FIG. 1

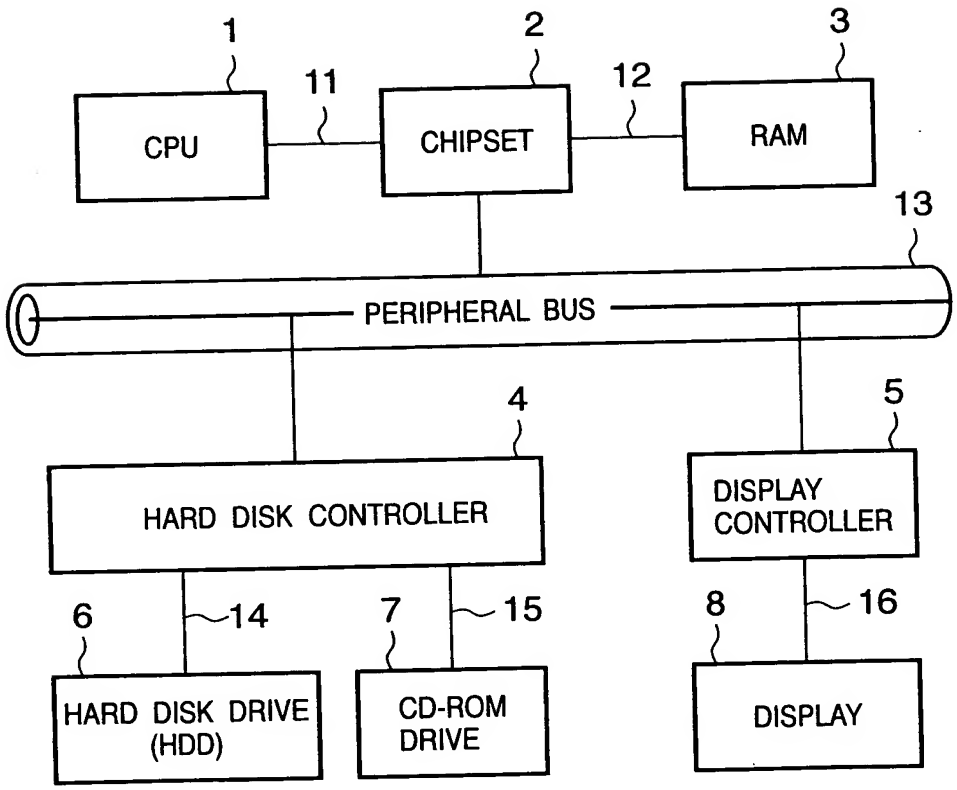
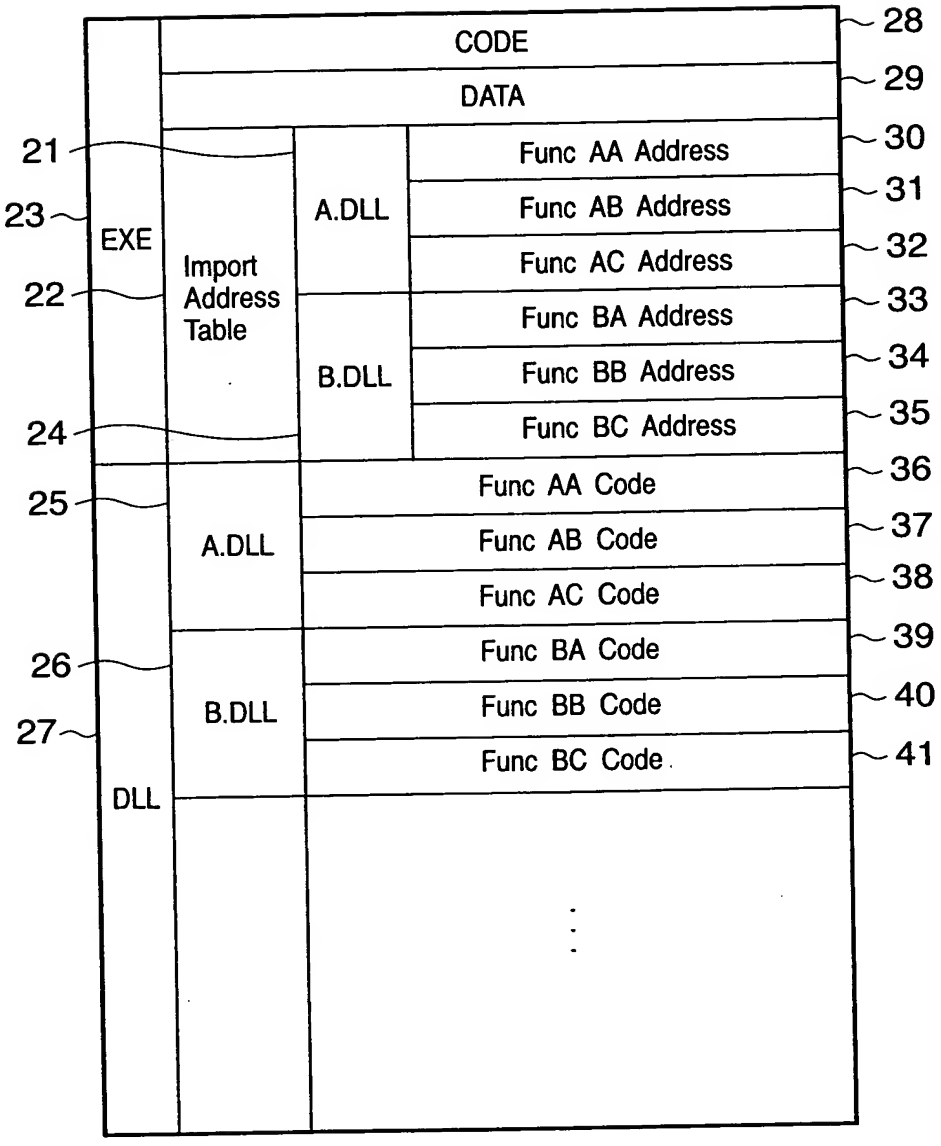

1
2
3
11
12
CPU
CHIPSET
RAM
13
PERIPHERAL BUS
4
5
HARD DISK CONTROLLER
DISPLAY CONTROLLER
6
14
7
15
8
16
HARD DISK DRIVE (HDD)
CD-ROM DRIVE
DISPLAY

FIG. 2


CODE
DATA
EXE
Import Address Table
A.DLL
B.DLL
Func AA Address
Func AB Address
Func AC Address
Func BA Address
Func BB Address
Func BC Address
DLL
A.DLL
B.DLL
Func AA Code
Func AB Code
Func AC Code
Func BA Code
Func BB Code
Func BC Code
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41

# FIG. 3

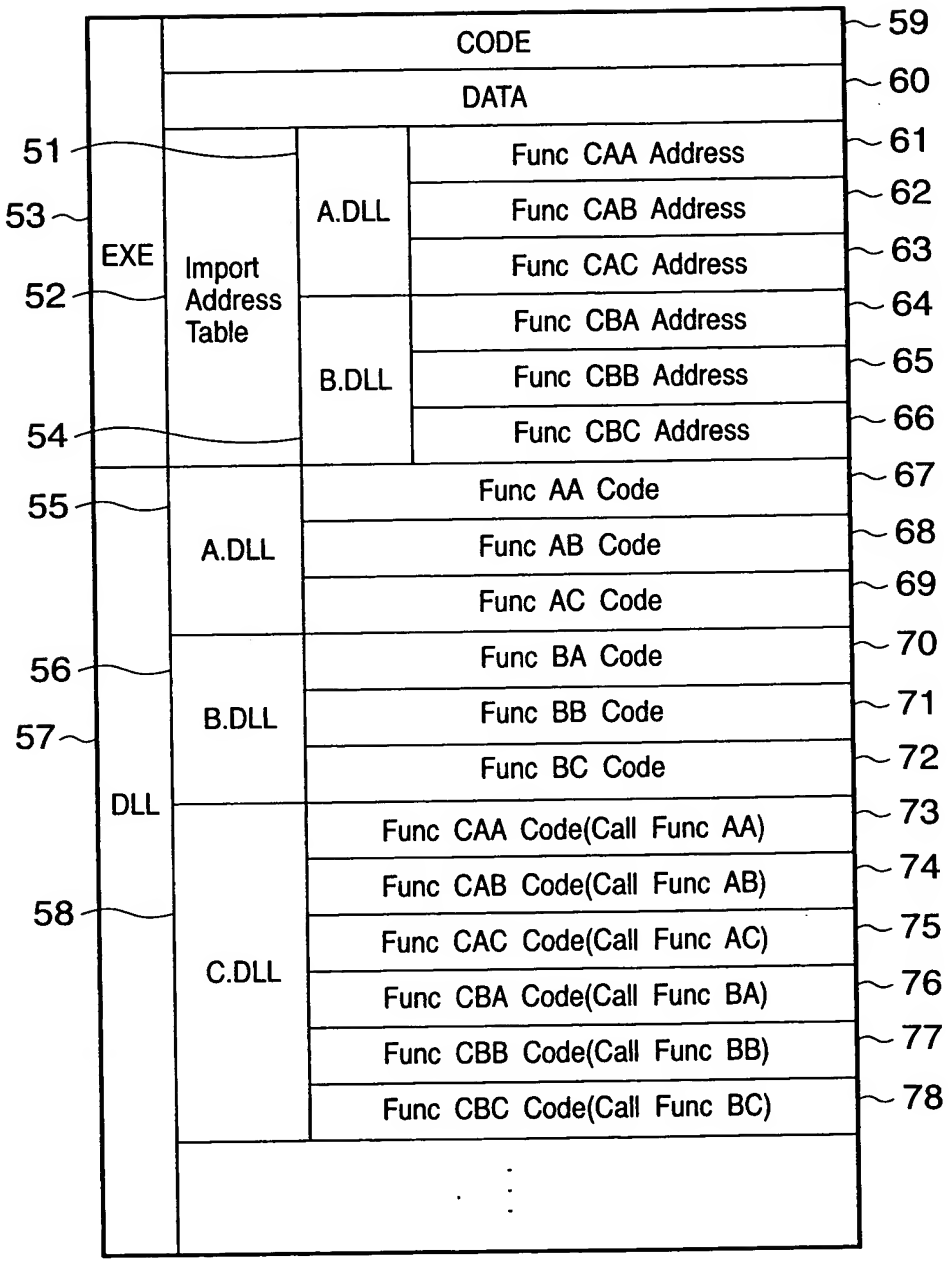

CODE
DATA
Func CAA Address
Func CAB Address
Func CAC Address
Func CBA Address
Func CBB Address
Func CBC Address
Func AA Code
Func AB Code
Func AC Code
Func BA Code
Func BB Code
Func BC Code
Func CAA Code(Call Func AA)
Func CAB Code(Call Func AB)
Func CAC Code(Call Func AC)
Func CBA Code(Call Func BA)
Func CBB Code(Call Func BB)
Func CBC Code(Call Func BC)
EXE
DLL
Import Address Table
A.DLL
B.DLL
A.DLL
B.DLL
C.DLL
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78

FIG. 4A

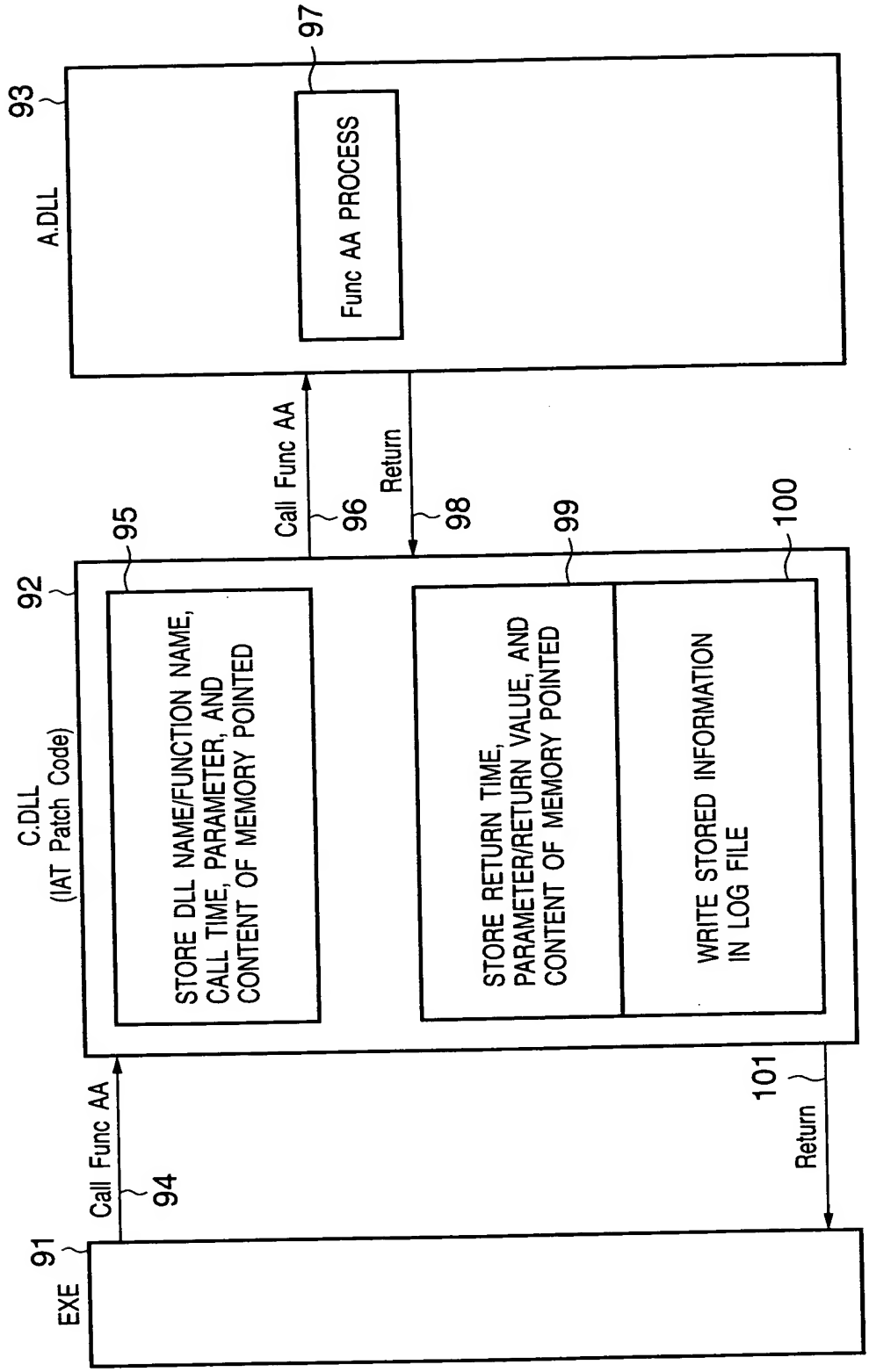

EXE
91
Call Func AA
94
C.DLL
(IAT Patch Code)
92
95
STORE DLL NAME/FUNCTION NAME, CALL TIME, PARAMETER, AND CONTENT OF MEMORY POINTED
Call Func AA
96
A.DLL
93
97
Func AA PROCESS
Return
98
99
STORE RETURN TIME, PARAMETER/RETURN VALUE, AND CONTENT OF MEMORY POINTED
100
WRITE STORED INFORMATION IN LOG FILE
101
Return

FIG. 4B

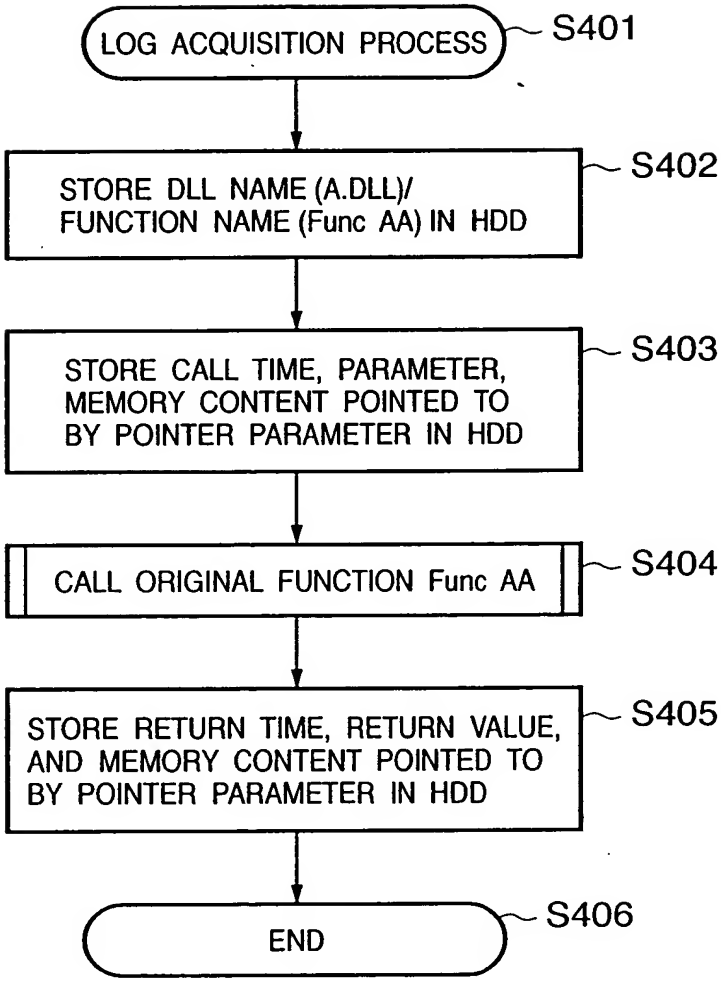
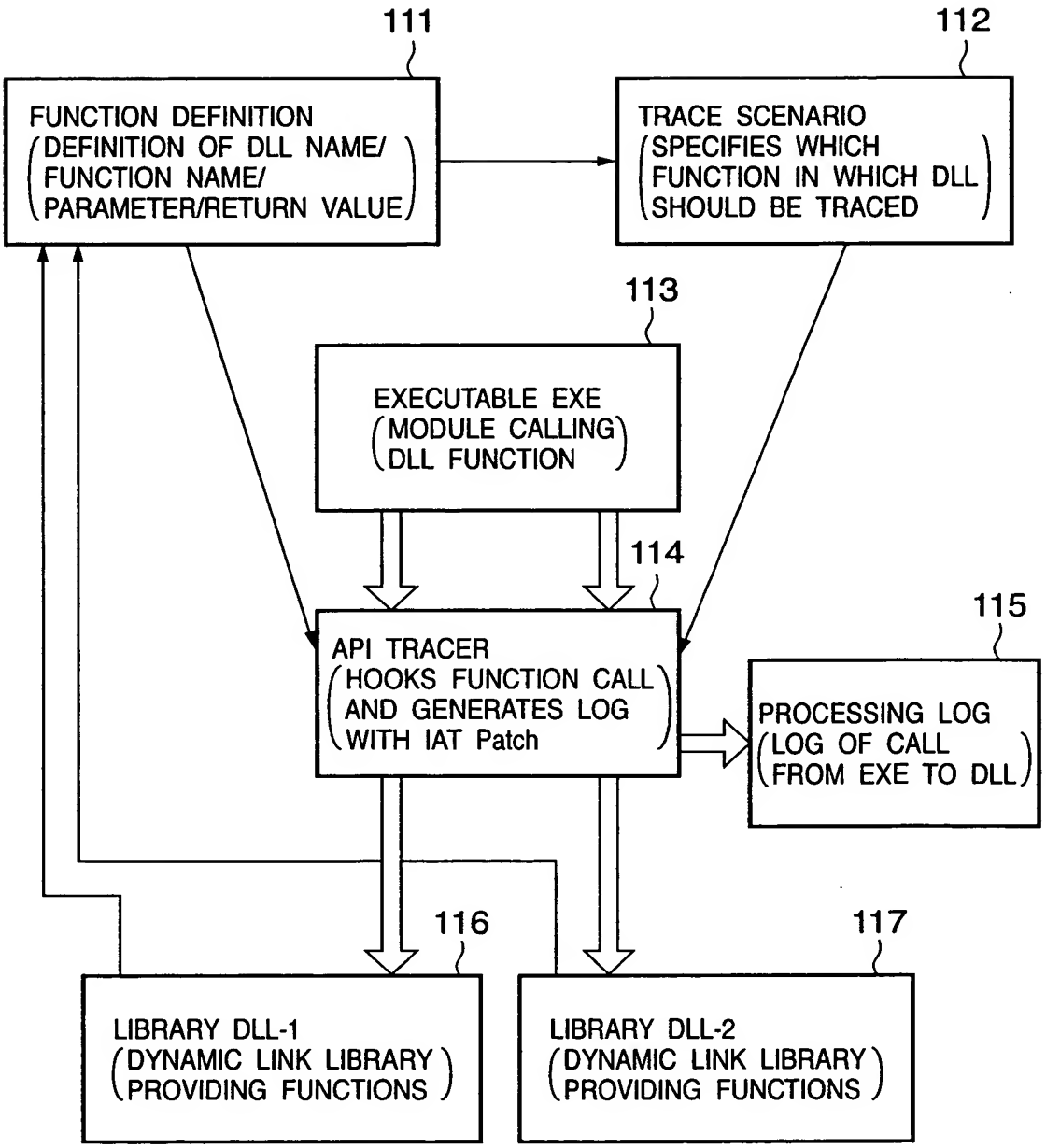
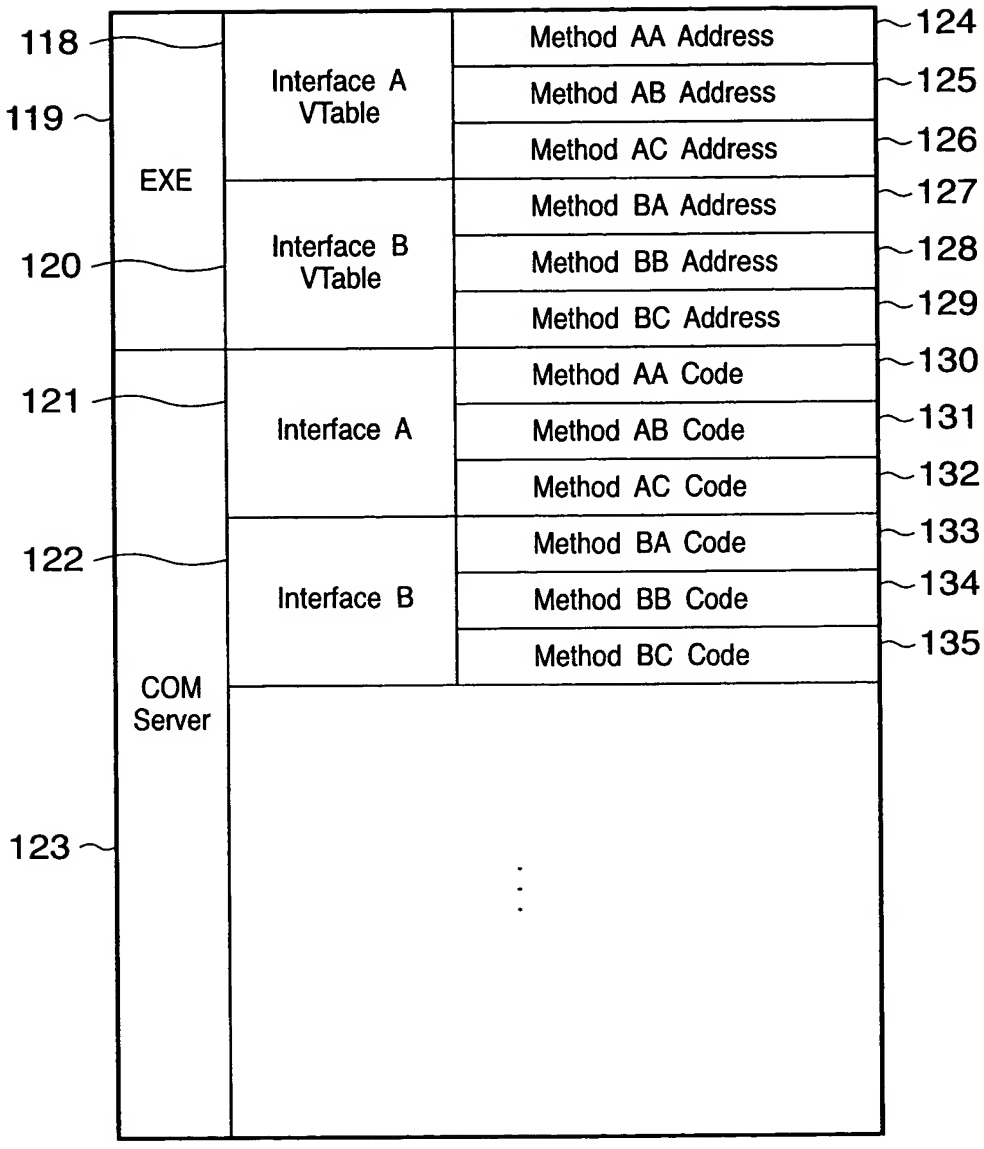

LOG ACQUISITION PROCESS
S401
STORE DLL NAME (A.DLL)/
FUNCTION NAME (Func AA) IN HDD
S402
STORE CALL TIME, PARAMETER,
MEMORY CONTENT POINTED TO
BY POINTER PARAMETER IN HDD
S403
CALL ORIGINAL FUNCTION Func AA
S404
STORE RETURN TIME, RETURN VALUE,
AND MEMORY CONTENT POINTED TO
BY POINTER PARAMETER IN HDD
S405
END
S406

FIG. 5


111
112
FUNCTION DEFINITION
(DEFINITION OF DLL NAME/
FUNCTION NAME/
PARAMETER/RETURN VALUE)
TRACE SCENARIO
(SPECIFIES WHICH
FUNCTION IN WHICH DLL
SHOULD BE TRACED)
113
EXECUTABLE EXE
(MODULE CALLING
DLL FUNCTION)
114
115
API TRACER
(HOOKS FUNCTION CALL
AND GENERATES LOG
WITH IAT Patch)
PROCESSING LOG
(LOG OF CALL
FROM EXE TO DLL)
116
117
LIBRARY DLL-1
(DYNAMIC LINK LIBRARY
PROVIDING FUNCTIONS)
LIBRARY DLL-2
(DYNAMIC LINK LIBRARY
PROVIDING FUNCTIONS)

# FIG. 6


118
119
120
121
122
123
EXE
COM Server
Interface A VTable
Interface B VTable
Interface A
Interface B
Method AA Address
Method AB Address
Method AC Address
Method BA Address
Method BB Address
Method BC Address
Method AA Code
Method AB Code
Method AC Code
Method BA Code
Method BB Code
Method BC Code
124
125
126
127
128
129
130
131
132
133
134
135

# FIG. 7

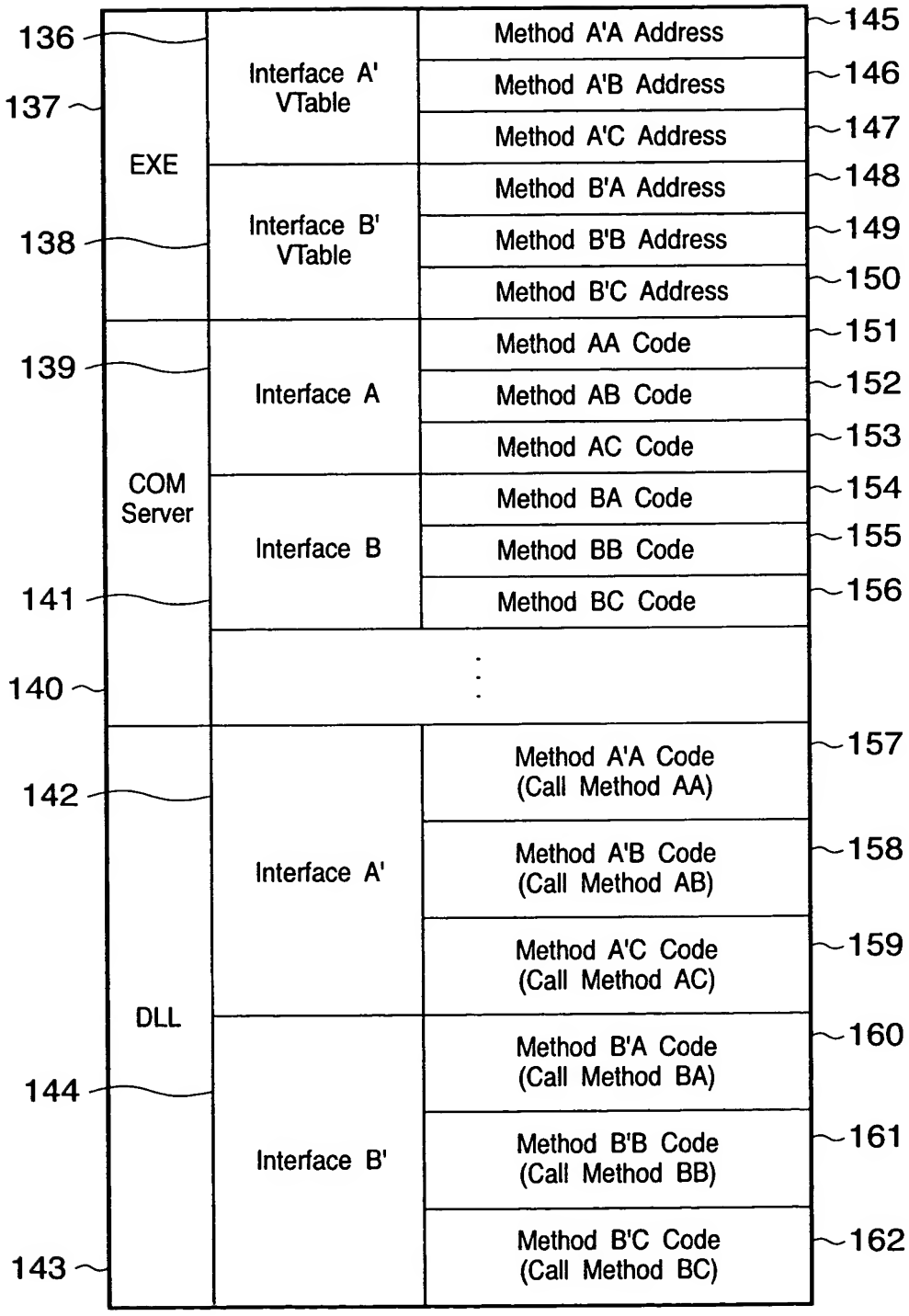

136
137
138
139
141
140
142
144
143
EXE
COM Server
DLL
Interface A' VTable
Interface B' VTable
Interface A
Interface B
Interface A'
Interface B'
Method A'A Address
Method A'B Address
Method A'C Address
Method B'A Address
Method B'B Address
Method B'C Address
Method AA Code
Method AB Code
Method AC Code
Method BA Code
Method BB Code
Method BC Code
Method A'A Code (Call Method AA)
Method A'B Code (Call Method AB)
Method A'C Code (Call Method AC)
Method B'A Code (Call Method BA)
Method B'B Code (Call Method BB)
Method B'C Code (Call Method BC)
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162

FIG. 8A

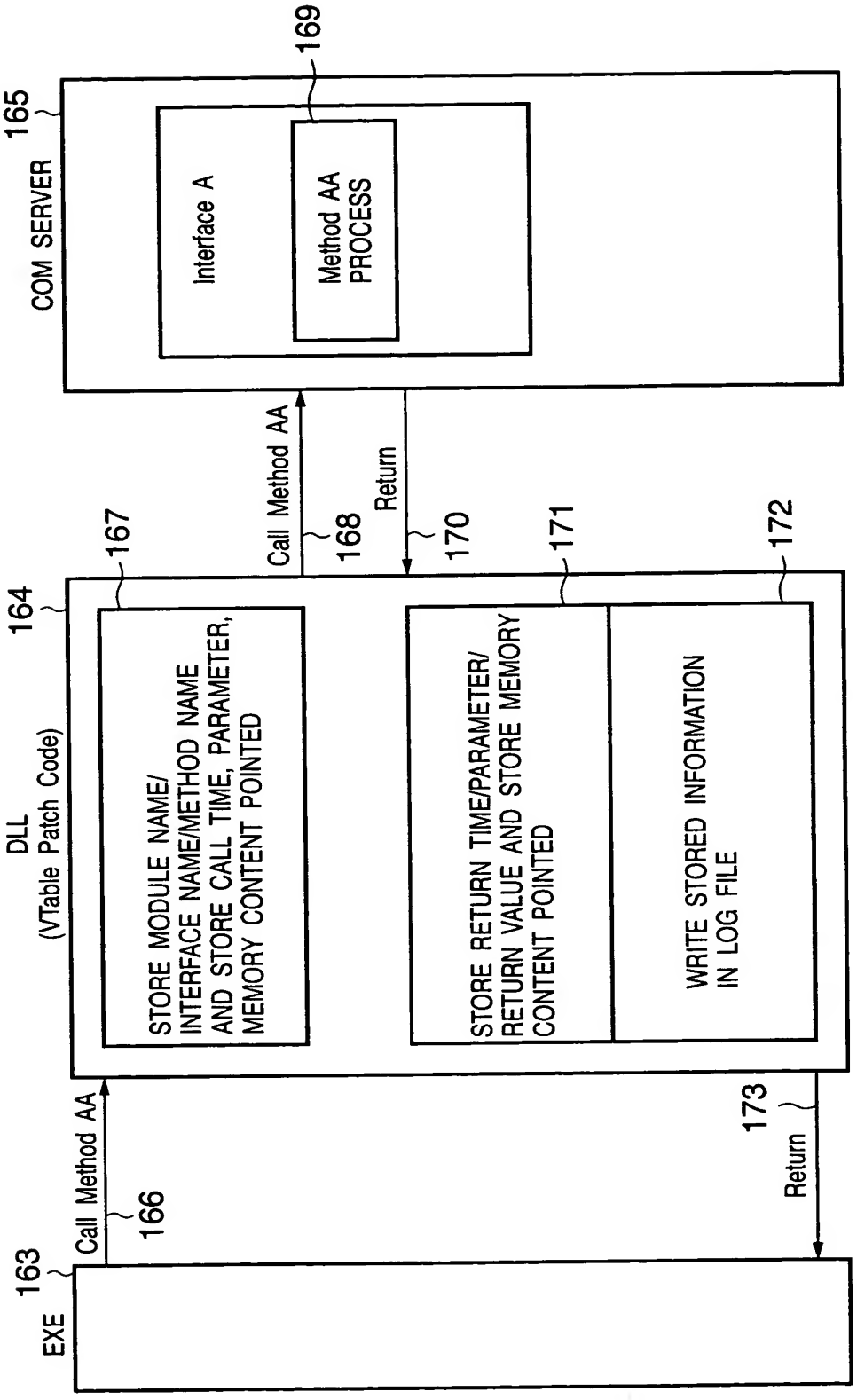
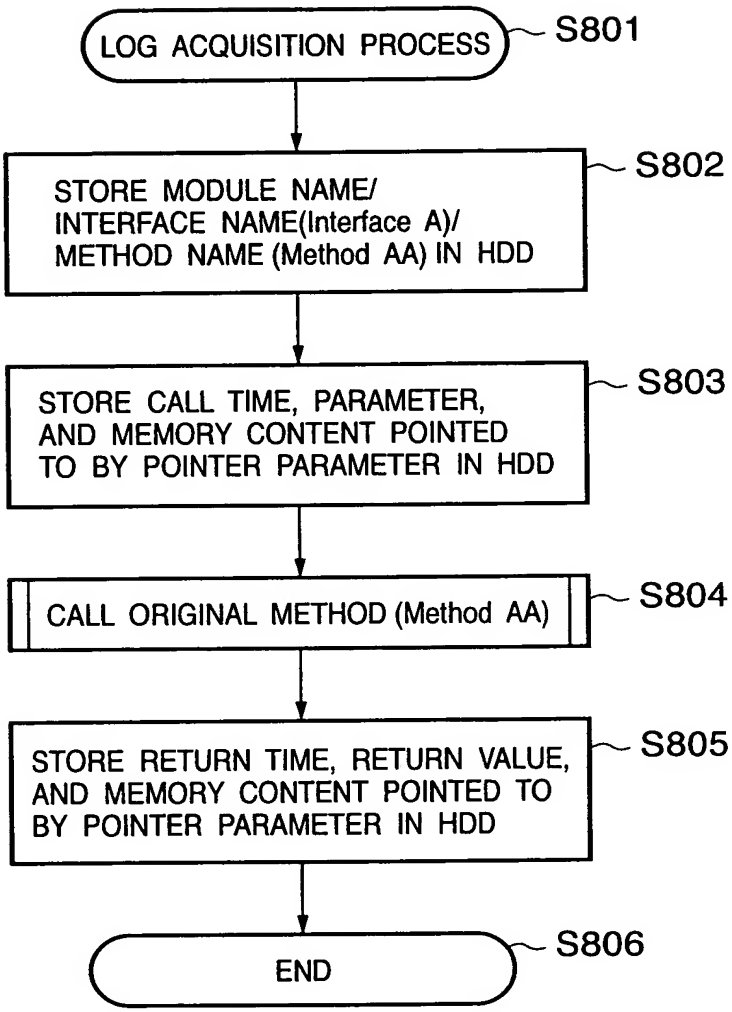
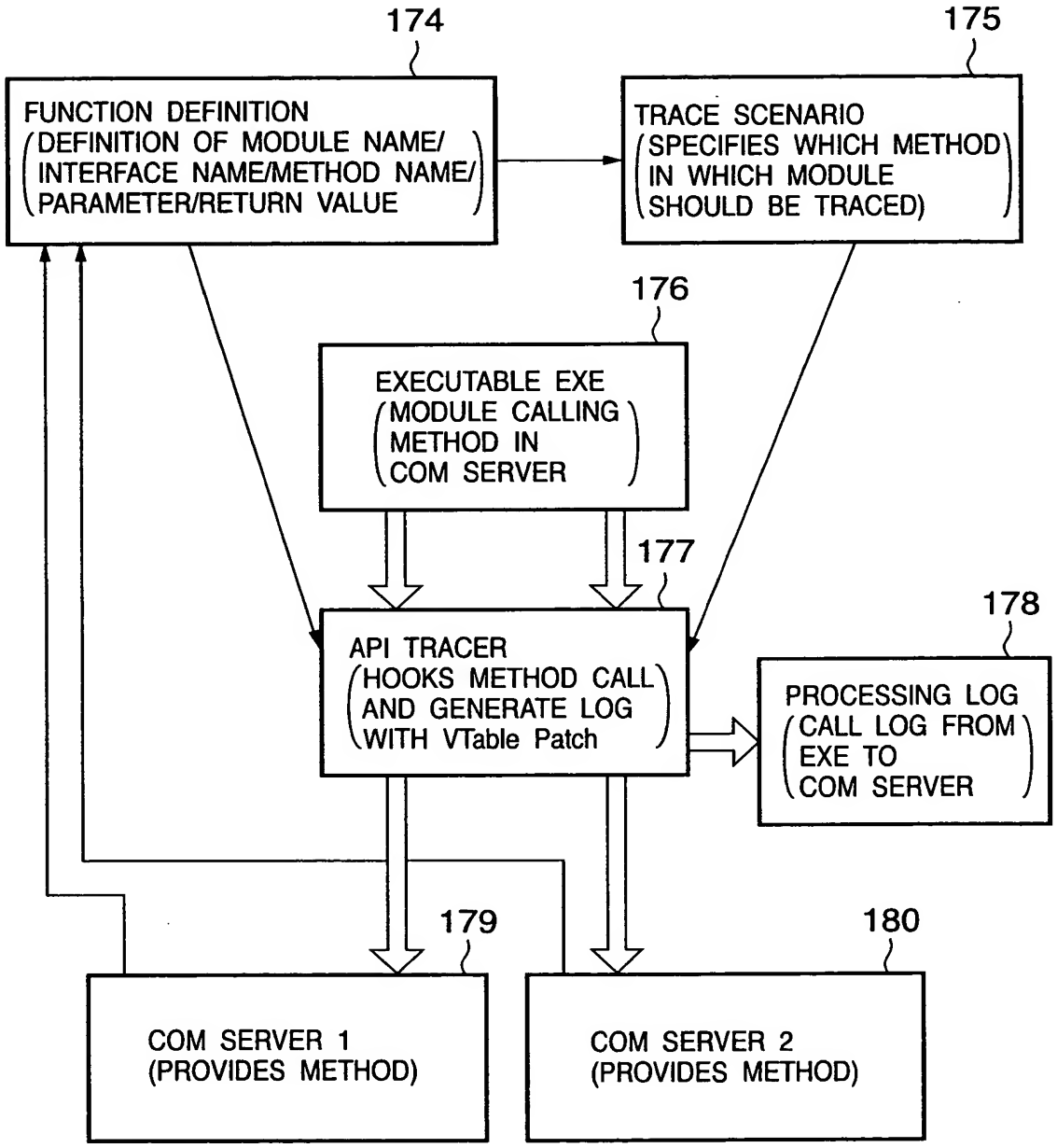

EXE
163
Call Method AA
166
DLL
(VTable Patch Code)
164
STORE MODULE NAME/
INTERFACE NAME/METHOD NAME
AND STORE CALL TIME, PARAMETER,
MEMORY CONTENT POINTED
167
Call Method AA
168
Return
170
STORE RETURN TIME/PARAMETER/
RETURN VALUE AND STORE MEMORY
CONTENT POINTED
171
WRITE STORED INFORMATION
IN LOG FILE
172
Return
173
COM SERVER
165
Interface A
Method AA
PROCESS
169

FIG. 8B


LOG ACQUISITION PROCESS
S801
STORE MODULE NAME/
INTERFACE NAME(Interface A)/
METHOD NAME (Method AA) IN HDD
S802
STORE CALL TIME, PARAMETER,
AND MEMORY CONTENT POINTED
TO BY POINTER PARAMETER IN HDD
S803
CALL ORIGINAL METHOD (Method AA)
S804
STORE RETURN TIME, RETURN VALUE,
AND MEMORY CONTENT POINTED TO
BY POINTER PARAMETER IN HDD
S805
END
S806

# FIG. 9


174
175
FUNCTION DEFINITION
(DEFINITION OF MODULE NAME/
INTERFACE NAME/METHOD NAME/
PARAMETER/RETURN VALUE
TRACE SCENARIO
(SPECIFIES WHICH METHOD
IN WHICH MODULE
SHOULD BE TRACED)
176
EXECUTABLE EXE
(MODULE CALLING
METHOD IN
COM SERVER
177
178
API TRACER
(HOOKS METHOD CALL
AND GENERATE LOG
WITH VTable Patch
PROCESSING LOG
(CALL LOG FROM
EXE TO
COM SERVER
179
180
COM SERVER 1
(PROVIDES METHOD)
COM SERVER 2
(PROVIDES METHOD)

# FIG. 10

```
[
      uuid(58DB5633-0694-4340-97CE-4E1AC6BFFBA7),      //TestDllStd
      helpstring("TestDllStd Type Library For PAT"),
      version(1,0)

]

library TestDllStd

        typedef [public] struct
        {
                char chParam;
                unsigned char uchParam;
                short sParam;
                unsigned short usParam;
                int nParam;
                unsigned int unParam;
                long lParam;
                unsigned long ulParam;
                double dbParam;
                float fParam;
        }TESTSTRUCT;
        typedef [public] TESTSTRUCT *LPTESTSTRUCT;
//DEFINE_GUID(GUID_PROGID, 0x8e037d65, 0xefa0, 0x40e7, 0x91, 0x43, 0xef, 0x70, 0x56,
0x94, 0x5b, 0x79);
  [

      uuid(8E037D65-EFA0-40e7-9143-EF7056945B79),
  ]   helpstring("TestDllStd.dll for PAT object"),

        interface
        test
        {
                char_stdcall FuncCharStd([in] char chPram);
                char*_stdcall FuncPCharStd([in, out] char* lpchParam);

                TESTSTRUCT_stdcall FuncStructStd[in]TESTSTRUCT TestStruct);
        };      LPTESTSTRUCT_stdcall FuncPStructStd([in, out]LPTESTSTRUCTlp TestStruct):
}
```

# FIG. 11

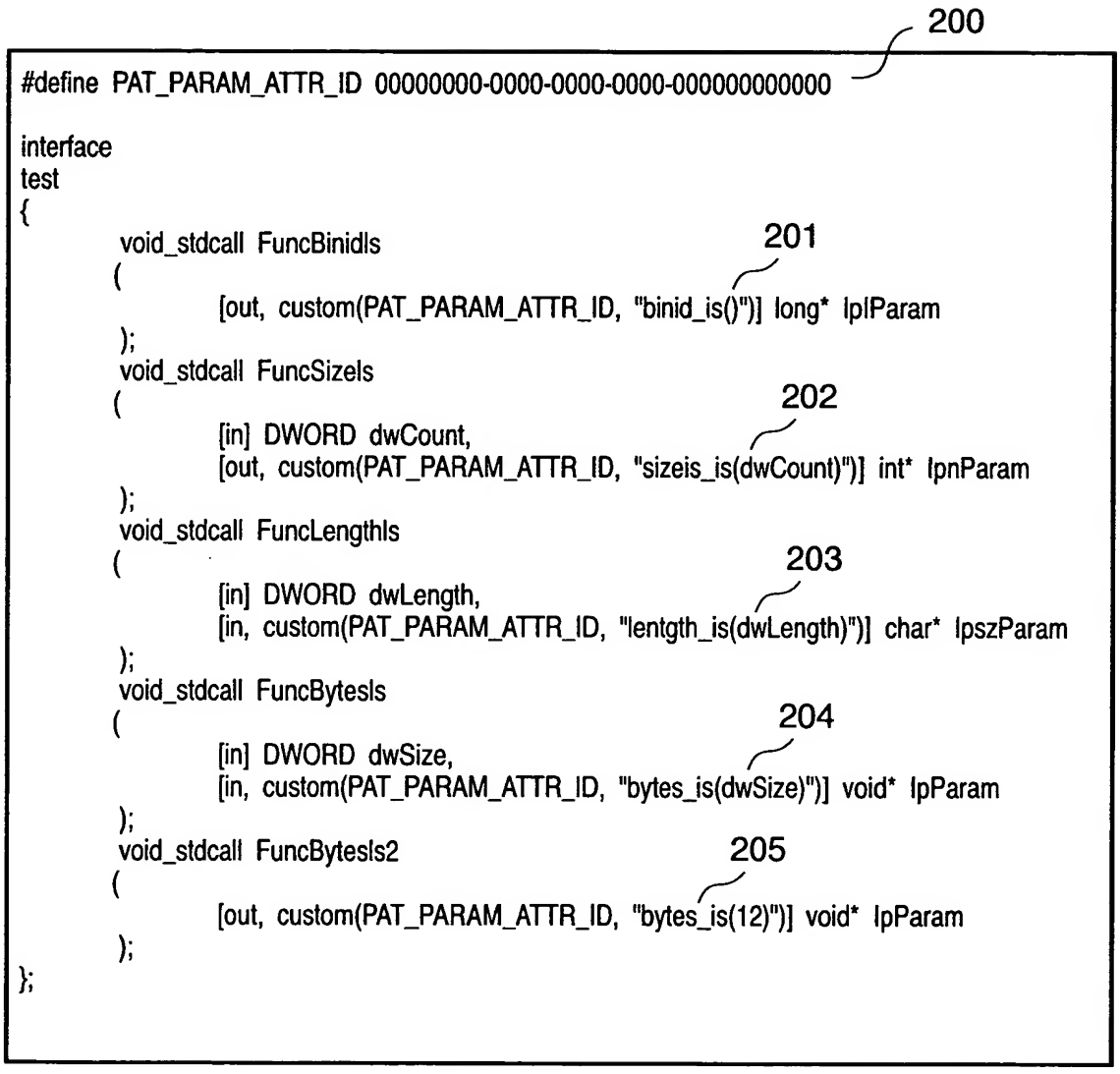

200
#define PAT_PARAM_ATTR_ID 00000000-0000-0000-0000-000000000000
interface
test
{
void_stdcall FuncBinidIs
201
(
[out, custom(PAT_PARAM_ATTR_ID, "binid_is()")] long* lplParam
);
void_stdcall FuncSizeIs
(
202
[in] DWORD dwCount,
[out, custom(PAT_PARAM_ATTR_ID, "sizeis_is(dwCount)")] int* lpnParam
);
void_stdcall FuncLengthIs
(
203
[in] DWORD dwLength,
[in, custom(PAT_PARAM_ATTR_ID, "lentgth_is(dwLength)")] char* lpszParam
);
void_stdcall FuncBytesIs
(
204
[in] DWORD dwSize,
[in, custom(PAT_PARAM_ATTR_ID, "bytes_is(dwSize)")] void* lpParam
);
void_stdcall FuncBytesIs2
205
(
[out, custom(PAT_PARAM_ATTR_ID, "bytes_is(12)")] void* lpParam
);
};

FIG. 12

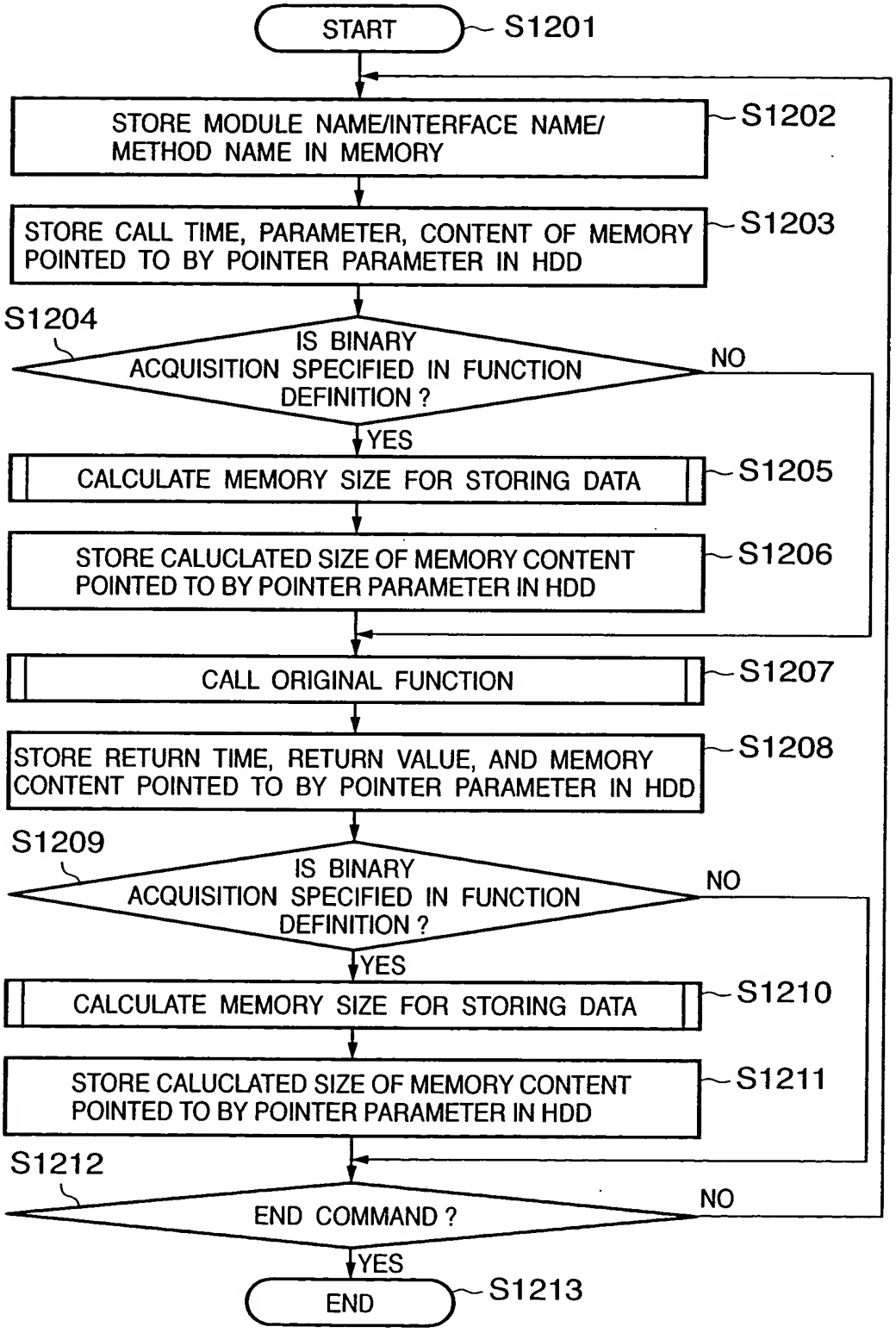
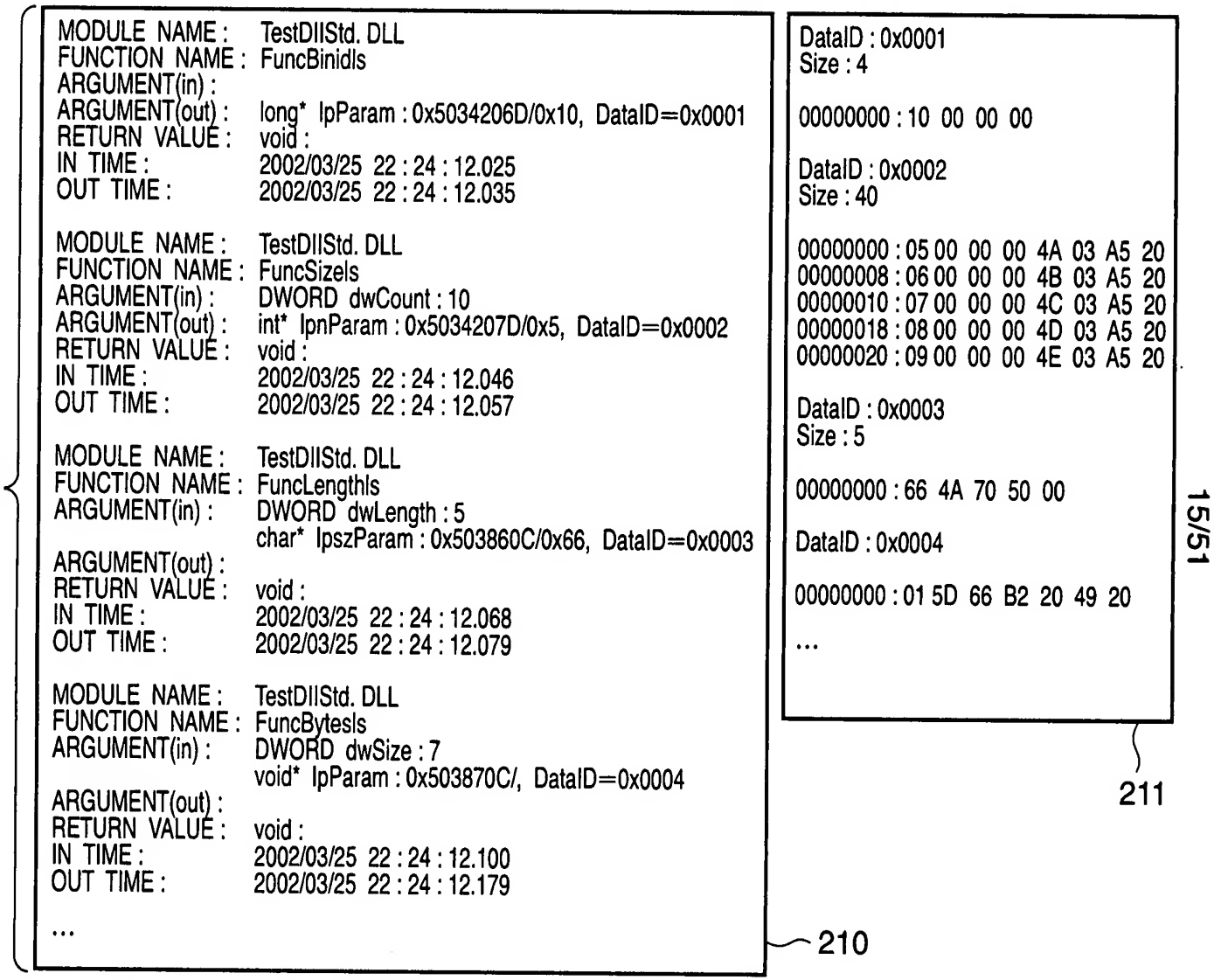

START
S1201
STORE MODULE NAME/INTERFACE NAME/ METHOD NAME IN MEMORY
S1202
STORE CALL TIME, PARAMETER, CONTENT OF MEMORY POINTED TO BY POINTER PARAMETER IN HDD
S1203
S1204
IS BINARY ACQUISITION SPECIFIED IN FUNCTION DEFINITION ?
NO
YES
CALCULATE MEMORY SIZE FOR STORING DATA
S1205
STORE CALUCLATED SIZE OF MEMORY CONTENT POINTED TO BY POINTER PARAMETER IN HDD
S1206
CALL ORIGINAL FUNCTION
S1207
STORE RETURN TIME, RETURN VALUE, AND MEMORY CONTENT POINTED TO BY POINTER PARAMETER IN HDD
S1208
S1209
IS BINARY ACQUISITION SPECIFIED IN FUNCTION DEFINITION ?
NO
YES
CALCULATE MEMORY SIZE FOR STORING DATA
S1210
STORE CALUCLATED SIZE OF MEMORY CONTENT POINTED TO BY POINTER PARAMETER IN HDD
S1211
S1212
END COMMAND ?
NO
YES
END
S1213

FIG. 13


MODULE NAME : TestDllStd. DLL
FUNCTION NAME : FuncBinidls
ARGUMENT(in) :
ARGUMENT(out) : long* lpParam : 0x5034206D/0x10, DataID=0x0001
RETURN VALUE : void :
IN TIME : 2002/03/25 22 : 24 : 12.025
OUT TIME : 2002/03/25 22 : 24 : 12.035
MODULE NAME : TestDllStd. DLL
FUNCTION NAME : FuncSizels
ARGUMENT(in) : DWORD dwCount : 10
ARGUMENT(out) : int* lpnParam : 0x5034207D/0x5, DataID=0x0002
RETURN VALUE : void :
IN TIME : 2002/03/25 22 : 24 : 12.046
OUT TIME : 2002/03/25 22 : 24 : 12.057
MODULE NAME : TestDllStd. DLL
FUNCTION NAME : FuncLengthls
ARGUMENT(in) : DWORD dwLength : 5
char* lpszParam : 0x503860C/0x66, DataID=0x0003
ARGUMENT(out) :
RETURN VALUE : void :
IN TIME : 2002/03/25 22 : 24 : 12.068
OUT TIME : 2002/03/25 22 : 24 : 12.079
MODULE NAME : TestDllStd. DLL
FUNCTION NAME : FuncBytesls
ARGUMENT(in) : DWORD dwSize : 7
void* lpParam : 0x503870C/, DataID=0x0004
ARGUMENT(out) :
RETURN VALUE : void :
IN TIME : 2002/03/25 22 : 24 : 12.100
OUT TIME : 2002/03/25 22 : 24 : 12.179
...
210
DataID : 0x0001
Size : 4
00000000 : 10 00 00 00
DataID : 0x0002
Size : 40
00000000 : 05 00 00 00 4A 03 A5 20
00000008 : 06 00 00 00 4B 03 A5 20
00000010 : 07 00 00 00 4C 03 A5 20
00000018 : 08 00 00 00 4D 03 A5 20
00000020 : 09 00 00 00 4E 03 A5 20
DataID : 0x0003
Size : 5
00000000 : 66 4A 70 50 00
DataID : 0x0004
00000000 : 01 5D 66 B2 20 49 20
...
211

# FIG. 14

```
#define PAT_PARAM_ATTR_ID 00000000-0000-0000-000000000000

typedef [public] struct ~ 220
{
    [in, custom(PAT_PARAM_ATTR_ID, "funcname_is(FuncInternal1)")] DWORD pfnFuncInternal1;
    [in, custom(PAT_PARAM_ATTR_ID, "funcname_is(FuncInternal2)")] DWORD pfnFuncInternal2;
    [in, custom(PAT_PARAM_ATTR_ID, "funcname_is(FuncInternal3)")] DWORD pfnFuncInternal3;
    [in, custom(PAT_PARAM_ATTR_ID, "funcname_is(FuncInternal4)")] DWORD pfnFuncInternal4;
}FUNCPOINTERARRAY;

interface
test
{
    void_stdcall SetCallBack                                                    221
    (
        [in, custom(PAT_PARAM_ATTR_ID, "funcname_is(FuncCallBack)")] DWORD
pfnFuncCallBack
    );
    void FuncCallBack([in] int nParam); ~ 222

    void_stdcall GetFuncPointer                                          223
    (
        [out, custom(PAT_PARAM_ATTR_ID, "funcname_is(FuncInternal)")] DWORD
pfnFuncInternal
    );
    void FuncInternal([in, out] char* lpszParam); ~ 224

    void_stdcall GetFuncPointerArray
    (
        [out]FUNCPOINTERARRAY* pFuncPointerArray; ~ 225
    );
    void FuncInternal1([in] int nParam);
    void FuncInternal2([in, out] char* lpzaParam);
    void FuncInternal3([out] DWORD* dwParam); ~ 226
    void FuncInternal4([);
};
```

# FIG. 15

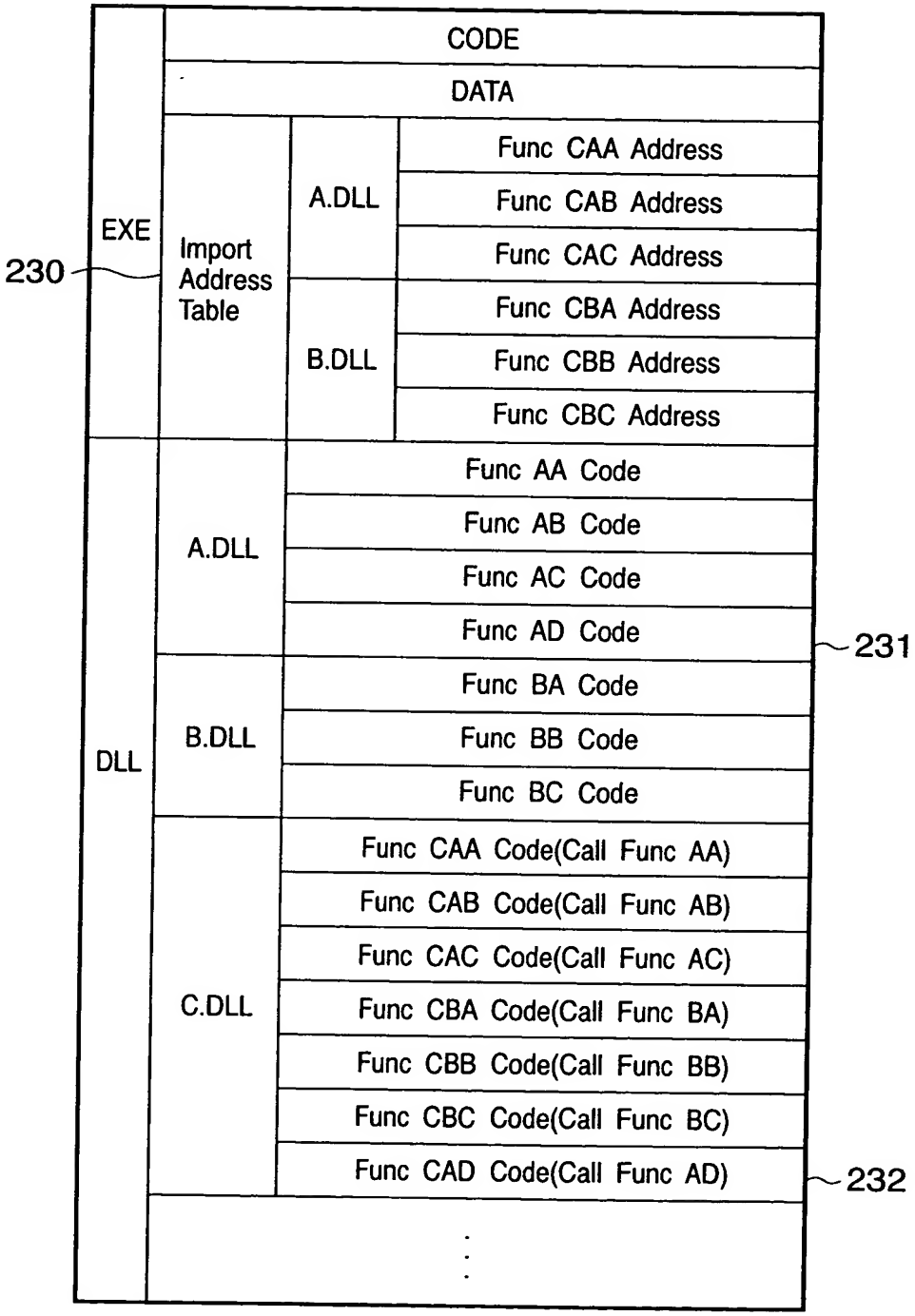
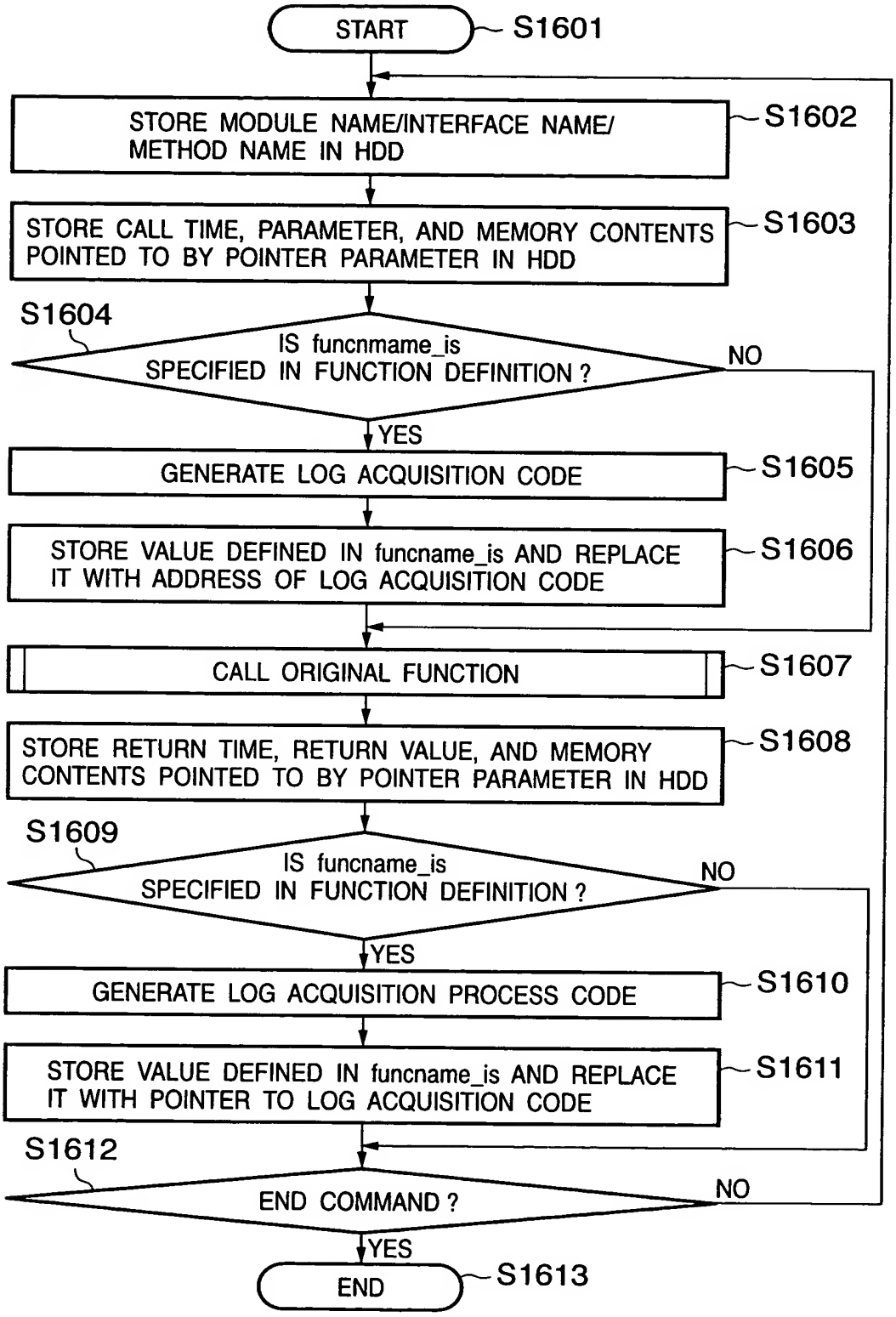

EXE
DLL
CODE
DATA
Import Address Table
230
A.DLL
B.DLL
C.DLL
Func CAA Address
Func CAB Address
Func CAC Address
Func CBA Address
Func CBB Address
Func CBC Address
Func AA Code
Func AB Code
Func AC Code
Func AD Code
Func BA Code
Func BB Code
Func BC Code
Func CAA Code(Call Func AA)
Func CAB Code(Call Func AB)
Func CAC Code(Call Func AC)
Func CBA Code(Call Func BA)
Func CBB Code(Call Func BB)
Func CBC Code(Call Func BC)
Func CAD Code(Call Func AD)
231
232

FIG. 16


START
S1601
STORE MODULE NAME/INTERFACE NAME/ METHOD NAME IN HDD
S1602
STORE CALL TIME, PARAMETER, AND MEMORY CONTENTS POINTED TO BY POINTER PARAMETER IN HDD
S1603
S1604
IS funcnmame_is SPECIFIED IN FUNCTION DEFINITION ?
NO
YES
GENERATE LOG ACQUISITION CODE
S1605
STORE VALUE DEFINED IN funcname_is AND REPLACE IT WITH ADDRESS OF LOG ACQUISITION CODE
S1606
CALL ORIGINAL FUNCTION
S1607
STORE RETURN TIME, RETURN VALUE, AND MEMORY CONTENTS POINTED TO BY POINTER PARAMETER IN HDD
S1608
S1609
IS funcname_is SPECIFIED IN FUNCTION DEFINITION ?
NO
YES
GENERATE LOG ACQUISITION PROCESS CODE
S1610
STORE VALUE DEFINED IN funcname_is AND REPLACE IT WITH POINTER TO LOG ACQUISITION CODE
S1611
S1612
END COMMAND ?
NO
YES
END
S1613

# FIG. 17

```
MODULE NAME :      TestDllStd. DLL
FUNCTION NAME :    SetCallBack
ARGUMENT(in) :     DWORD pfnFuncCallBack : 0x0299103F
ARGUMENT(out) :
RETURN VALUE :     void :
IN TIME :          2002/03/25 22 : 24 : 12.025
OUT TIME :         2002/03/25 22 : 24 : 12.035

MODULE NAME :      TestDllStd. DLL
FUNCTION NAME :    GetFuncPointer
ARGUMENT(in) :
ARGUMENT(out) :    DWORD pfnFuncInternal : 0x29913dF
RETURN VALUE :     void :
IN TIME :          2002/03/25 22 : 24 : 12.046
OUT TIME :         2002/03/25 22 : 24 : 12.057

MODULE NAME :      TestDllStd. DLL
FUNCTION NAME :    GetFuncPointerArray
ARGUMENT(in) :
ARGUMENT(out) :    FUNCPOINTERARRAY* pFuncPointerArray : 0x503860C
                   DWORD FUNCPOINTERARRAY. pfnFuncInternal1 : 0x02997670
                   DWORD FUNCPOINTERARRAY. pfnFuncInternal2 : 0x02997708
                   DWORD FUNCPOINTERARRAY. pfnFuncInternal3 : 0x029977BE
                   DWORD FUNCPOINTERARRAY. pfnFuncInternal4 : 0x0299784F
RETURN VALUE :     void :
IN TIME :          2002/03/25 22 : 24 : 12.068
OUT TIME :         2002/03/25 22 : 24 : 12.079
```

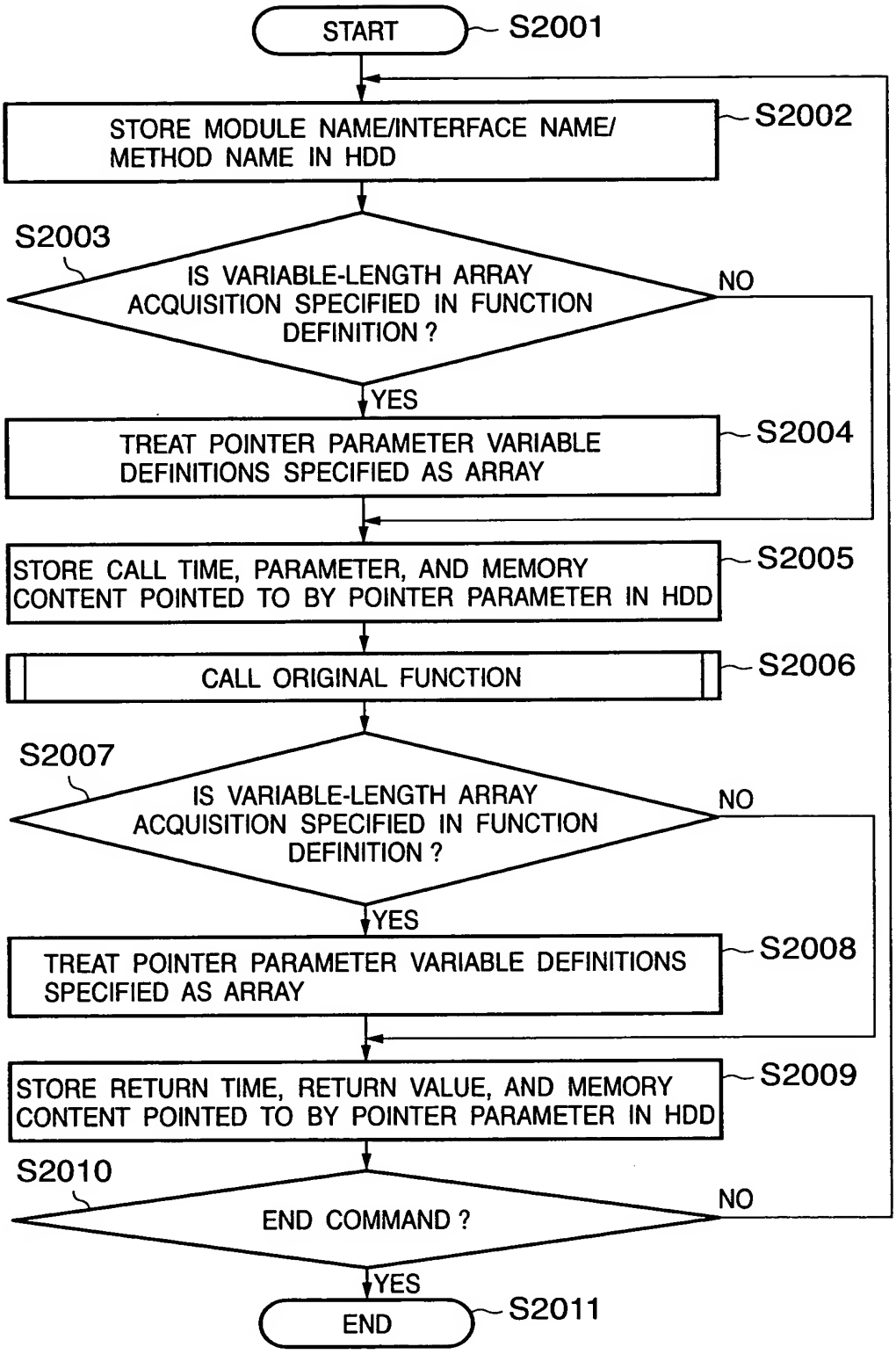
# FIG. 18

```
MODULE NAME :       TestDllStd. DLL
FUNCTION NAME :     SetCallBack
ARGUMENT(in) :      DWORD pfnFuncCallBack : 0x0299103F
ARGUMENT(out) :
RETURN VALUE :      void :
IN TIME :           2002/03/25 22 : 24 : 12.025
OUT TIME :          2002/03/25 22 : 24 : 12.035

MODULE NAME :       TestDllStd. DLL
FUNCTION NAME :     FuncCallBack
ARGUMENT(in) :      int nParam : 100
ARGUMENT(out) :
RETURN VALUE :      void :
IN TIME :           2002/03/25 22 : 24 : 12.036
OUT TIME :          2002/03/25 22 : 24 : 12.040

MODULE NAME :       TestDllStd. DLL
FUNCTION NAME :     GetFuncPointer
ARGUMENT(in) :
ARGUMENT(out) :     DWORD pfnFuncInternal : 0x029913dF
RETURN VALUE :      void :
IN TIME :           2002/03/25 22 : 24 : 12.046
OUT TIME :          2002/03/25 22 : 24 : 12.057

MODULE NAME :       TestDllStd. DLL
FUNCTION NAME :     FuncInternal1
ARGUMENT(in) :      char* lpszParam : 0x5038600/0
ARGUMENT(out) :     char* lpszParam : 0x5038600/−12
RETURN VALUE :      void :
IN TIME :           2002/03/25 22 : 24 : 12.060
OUT TIME :          2002/03/25 22 : 24 : 12.065

MODULE NAME :       TestDllStd. DLL
FUNCTION NAME :     GetFuncPointArray
ARGUMENT(in) :
ARGUMENT(out) :     FUNCPOINTERARRAY* pFuncPointerArray : 0x503860C
                    DWORD FUNCPOINTERARRAY. pfnFuncInternal1 : 0x02997670
                    DWORD FUNCPOINTERARRAY. pfnFuncInternal2 : 0x02997708
                    DWORD FUNCPOINTERARRAY. pfnFuncInternal3 : 0x029977BE
                    DWORD FUNCPOINTERARRAY. pfnFuncInternal4 : 0x0299784F
RETURN VALUE :      void :
IN TIME :           2002/03/25 22 : 24 : 12.068
OUT TIME :          2002/03/25 22 : 24 : 12.079

MODULE NAME :       TestDllStd. DLL
FUNCTION NAME :     FuncInternal4
ARGUMENT(in) :
ARGUMENT(out) :
RETURN VALUE :      void :
IN TIME :           2002/03/25 22 : 24 : 12.080
OUT TIME :          2002/03/25 22 : 24 : 12.099
```

# FIG. 19

```
#define PAT_PARAM_ATTR_ID 00000000-0000-0000-000000000000

interface
test
{
        void_stdcall FuncArrayIs
        (                                                                    240
                [in] DWORD dwCount,
                [in, out, custom(PAT_PARAM_ATTR_ID, "array_is(dwCount)")] int* lpnParam
        );
};
```

FIG. 20


START
S2001
STORE MODULE NAME/INTERFACE NAME/ METHOD NAME IN HDD
S2002
S2003
IS VARIABLE-LENGTH ARRAY ACQUISITION SPECIFIED IN FUNCTION DEFINITION ?
NO
YES
TREAT POINTER PARAMETER VARIABLE DEFINITIONS SPECIFIED AS ARRAY
S2004
STORE CALL TIME, PARAMETER, AND MEMORY CONTENT POINTED TO BY POINTER PARAMETER IN HDD
S2005
CALL ORIGINAL FUNCTION
S2006
S2007
IS VARIABLE-LENGTH ARRAY ACQUISITION SPECIFIED IN FUNCTION DEFINITION ?
NO
YES
TREAT POINTER PARAMETER VARIABLE DEFINITIONS SPECIFIED AS ARRAY
S2008
STORE RETURN TIME, RETURN VALUE, AND MEMORY CONTENT POINTED TO BY POINTER PARAMETER IN HDD
S2009
S2010
END COMMAND ?
NO
YES
END
S2011

# FIG. 21

```
MODULE NAME :    TestDllStd. DLL
FUNCTION NAME :  FuncArrayIs
ARGUMENT(in) :   DWORD dwCount : 4
                 int* lpnParam : 0x5034206D/0x00
ARGUMENT(out) :  int* lpnParam : 0x5034206D/0x01
RETURN VALUE :   void :
IN TIME :        2002/03/25 22 : 24 : 12.025
OUT TIME :       2002/03/25 22 : 24 : 12.035

MODULE NAME :    TestDllStd. DLL
FUNCTION NAME :  FuncArrayIs
ARGUMENT(in) :   DWORD dwCount : 3
                 int* lpnParam : 0x5034207D/0x00
ARGUMENT(out) :  int* lpnParam : 0x5034207D/0x05
RETURN VALUE :   void :
IN TIME :        2002/03/25 22 : 24 : 12.046
OUT TIME :       2002/03/25 22 : 24 : 12.057

...
```

250

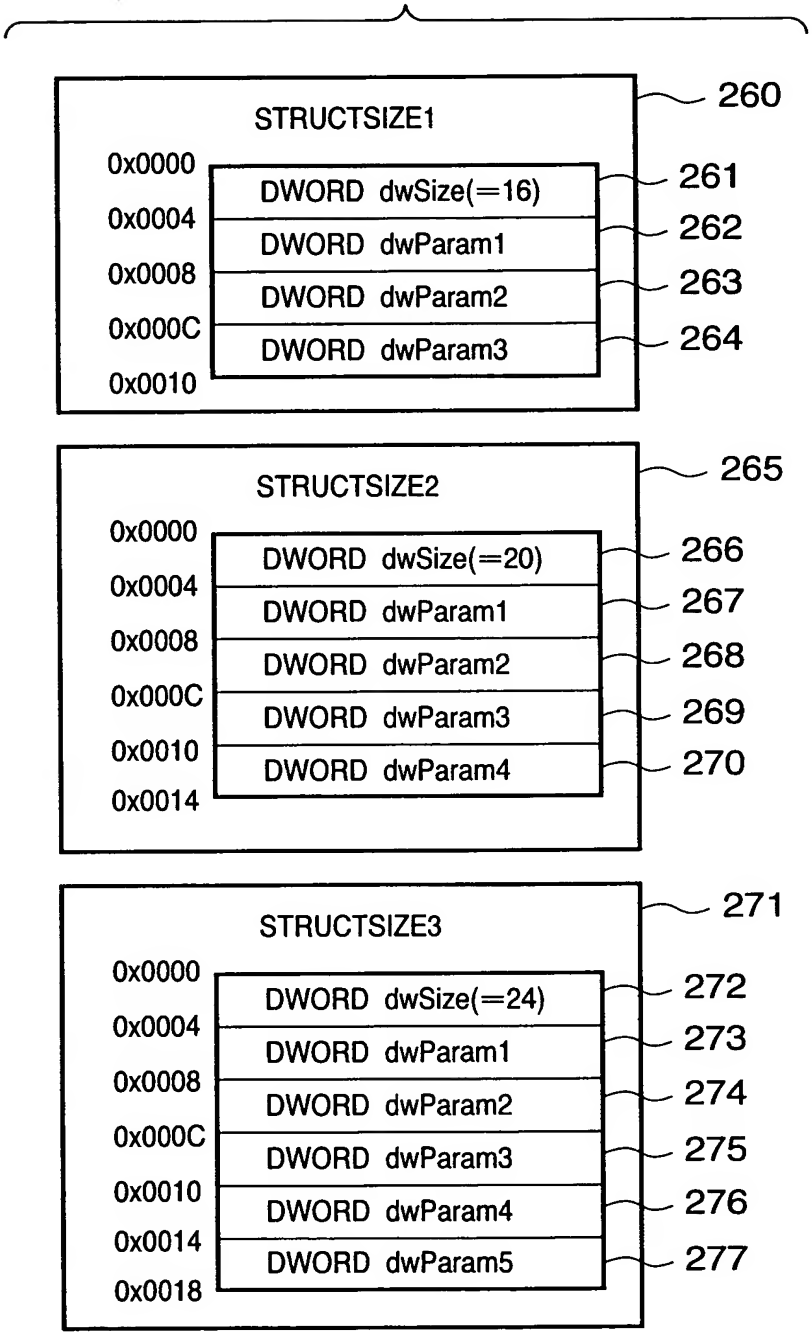
251

```
MODULE NAME :    TestDllStd. DLL
FUNCTION NAME :  FuncArrayIs
ARGUMENT(in) :   DWORD dwCount : 4
                 int* lpnParam : 0x5034206D/Array (int : 0 : 0x00, int : 1 : 0x00, int : 2 : 0x00, int : 3 : 0x00)
ARGUMENT(out) :  int* lpnParam : 0x5034206D/Array (int : 0 : 0x01, int : 1 : 0x02, int : 2 : 0x03, int : 3 : 0x04)
RETURN VALUE :   void :
IN TIME :        2002/03/25 22 : 24 : 12.025
OUT TIME :       2002/03/25 22 : 24 : 12.035

MODULE NAME :    TestDllStd. DLL
FUNCTION NAME :  FuncArrayIs
ARGUMENT(in) :   DWORD dwCount : 3
                 int* lpnParam : 0x5034207D/Array (int : 0 : 0x00, int : 1 : 0x00, int : 2 : 0x00)
ARGUMENT(out) :  int* lpnParam : 0x5034207D/Array (int : 0 : 0x05, int : 1 : 0x10, int : 2 : 0x15)
RETURN VALUE :   void :
IN TIME :        2002/03/25 22 : 24 : 12.046
OUT TIME :       2002/03/25 22 : 24 : 12.057

...
```

# FIG. 22

```
typedef struct
{
        DWORD dwSize ;
        DWORD dwParam1 ;
        DWORD dwParam2 ;
        DWORD dwParam3 ;
}STRUCTSIZE1 ;

typedef struct
{
        DWORD dwSize ;
        DWORD dwParam1 ;
        DWORD dwParam2 ;
        DWORD dwParam3 ;
        DWORD dwParam4 ;
}STRUCTSIZE2 ;

typedef struct
{
        DWORD dwSize ;
        DWORD dwParam1 ;
        DWORD dwParam2 ;
        DWORD dwParam3 ;
        DWORD dwParam4 ;
        DWORD dwParam5 ;
}STRUCTSIZE3 ;

void FuncGetData (DWORD dwKind, void* lpBuf)
{
        switch(dwKind)
        }
        case 1 :
                //lpBuf IS TREATED AS THE POINTER TO STRUCTSIZE1
                break;
        case 2 :
                //lpBuf IS TREATED AS THE POINTER TO STRUCTSIZE2
                break;
        case 3 :
                //lpBuf IS TREATED AS THE POINTER TO STRUCTSIZE3
                break;
        }
}
```

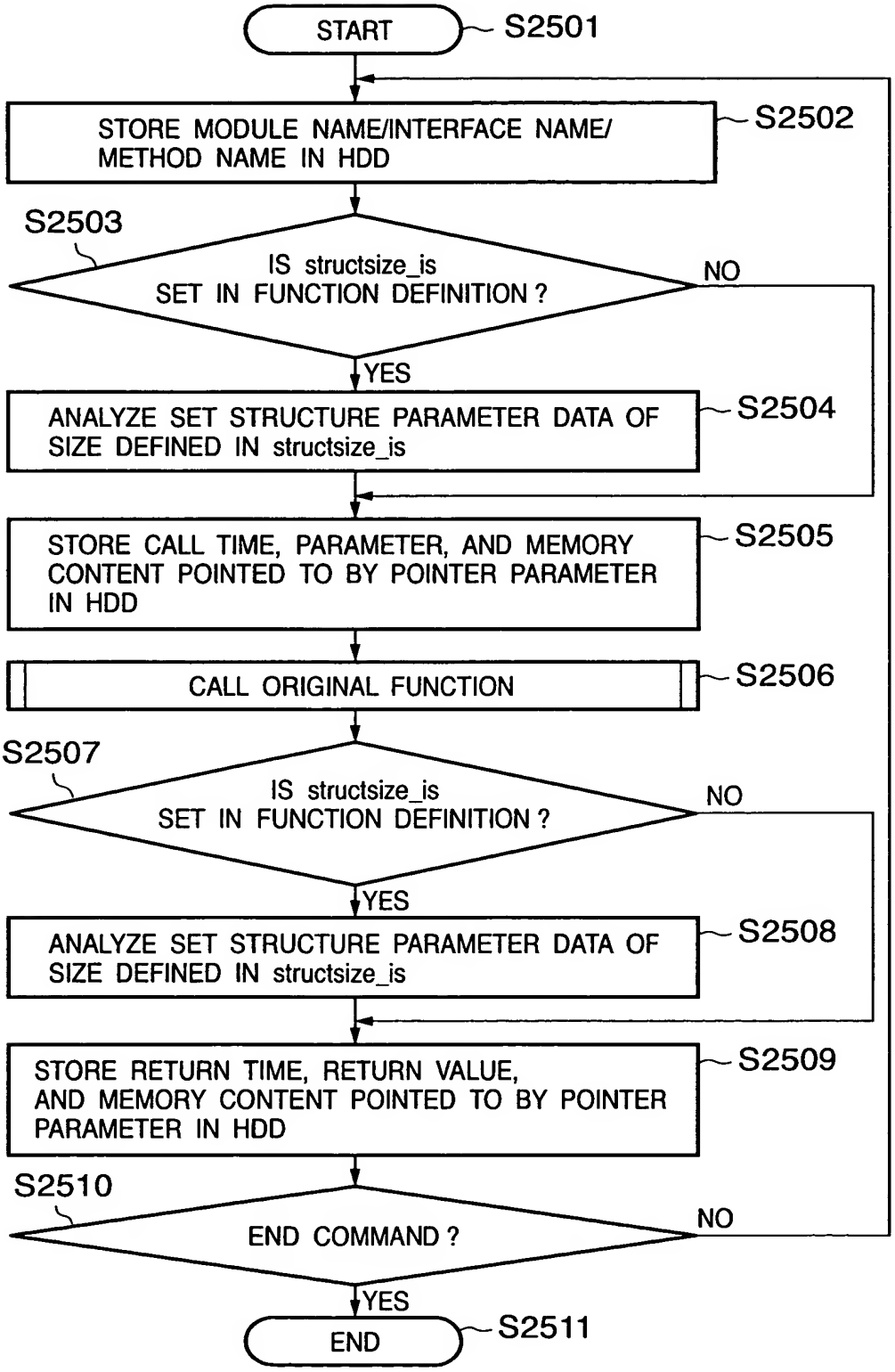
FIG. 23


STRUCTSIZE1
260
0x0000
DWORD dwSize(=16)
261
0x0004
DWORD dwParam1
262
0x0008
DWORD dwParam2
263
0x000C
DWORD dwParam3
264
0x0010
STRUCTSIZE2
265
0x0000
DWORD dwSize(=20)
266
0x0004
DWORD dwParam1
267
0x0008
DWORD dwParam2
268
0x000C
DWORD dwParam3
269
0x0010
DWORD dwParam4
270
0x0014
STRUCTSIZE3
271
0x0000
DWORD dwSize(=24)
272
0x0004
DWORD dwParam1
273
0x0008
DWORD dwParam2
274
0x000C
DWORD dwParam3
275
0x0010
DWORD dwParam4
276
0x0014
DWORD dwParam5
277
0x0018

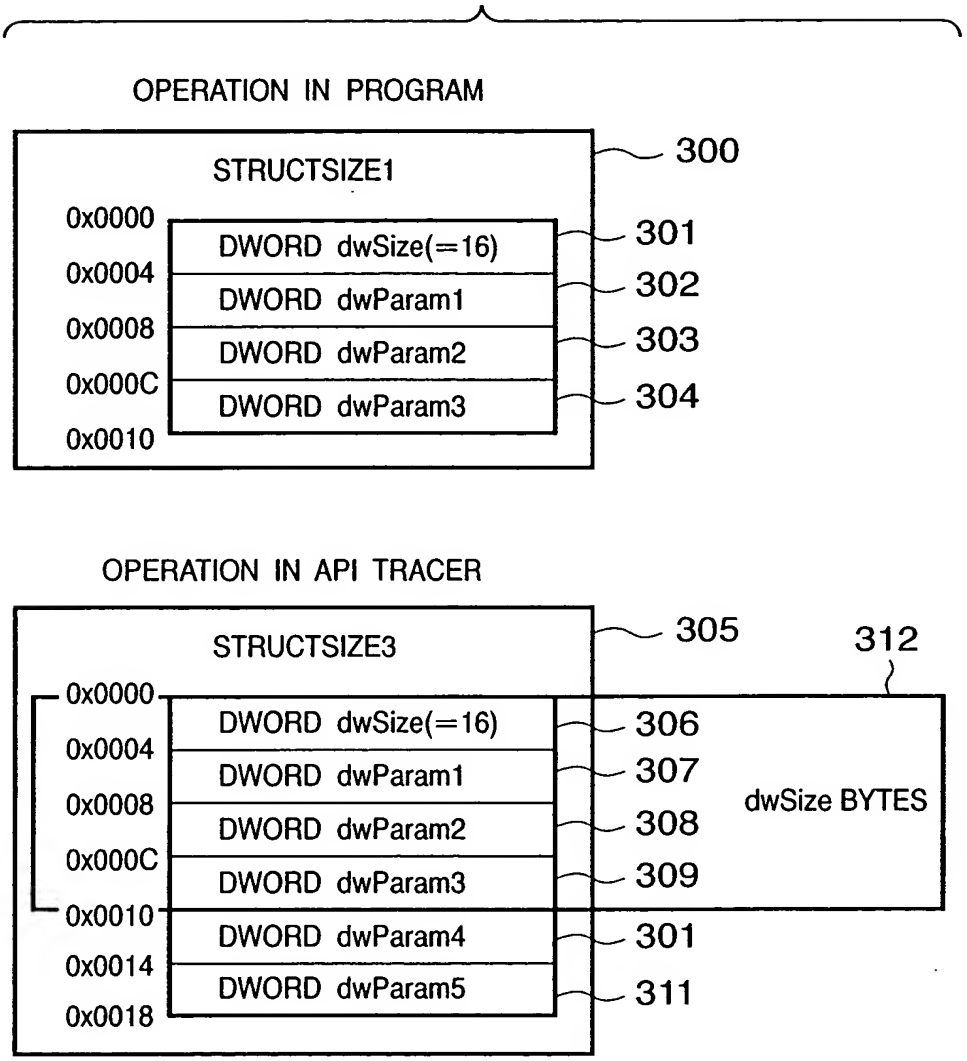
# FIG. 24

```
#define PAT PARAM_ATTR_ID 00000000-0000-0000-0000-000000000000

typedef [public] struct                                          290
{
        [custom (PAT_PARAM_ATTR_ID, "structsize_is()")]DWORD dwSize;
        DWORD dwParam1 ;
        DWORD dwParam2 ;
        DWORD dwParam3 ;
        DWORD dwParam4 ;
        DWORD dwParam5 ;

}STRUCTSIZE ;  ~ 291

interface
test
{
        void FuncGetData
        (                                                        292
                [in] DWORD dwKind,
                [in, out] STRUCTSIZE* lpBuf
        );

};
```

FIG. 25


START
S2501
STORE MODULE NAME/INTERFACE NAME/ METHOD NAME IN HDD
S2502
S2503
IS structsize_is SET IN FUNCTION DEFINITION ?
NO
YES
ANALYZE SET STRUCTURE PARAMETER DATA OF SIZE DEFINED IN structsize_is
S2504
STORE CALL TIME, PARAMETER, AND MEMORY CONTENT POINTED TO BY POINTER PARAMETER IN HDD
S2505
CALL ORIGINAL FUNCTION
S2506
S2507
IS structsize_is SET IN FUNCTION DEFINITION ?
NO
YES
ANALYZE SET STRUCTURE PARAMETER DATA OF SIZE DEFINED IN structsize_is
S2508
STORE RETURN TIME, RETURN VALUE, AND MEMORY CONTENT POINTED TO BY POINTER PARAMETER IN HDD
S2509
S2510
END COMMAND ?
NO
YES
END
S2511

# FIG. 26


OPERATION IN PROGRAM
STRUCTSIZE1
300
0x0000
0x0004
0x0008
0x000C
0x0010
DWORD dwSize(=16)
DWORD dwParam1
DWORD dwParam2
DWORD dwParam3
301
302
303
304
OPERATION IN API TRACER
STRUCTSIZE3
305
312
0x0000
0x0004
0x0008
0x000C
0x0010
0x0014
0x0018
DWORD dwSize(=16)
DWORD dwParam1
DWORD dwParam2
DWORD dwParam3
DWORD dwParam4
DWORD dwParam5
306
307
308
309
301
311
dwSize BYTES

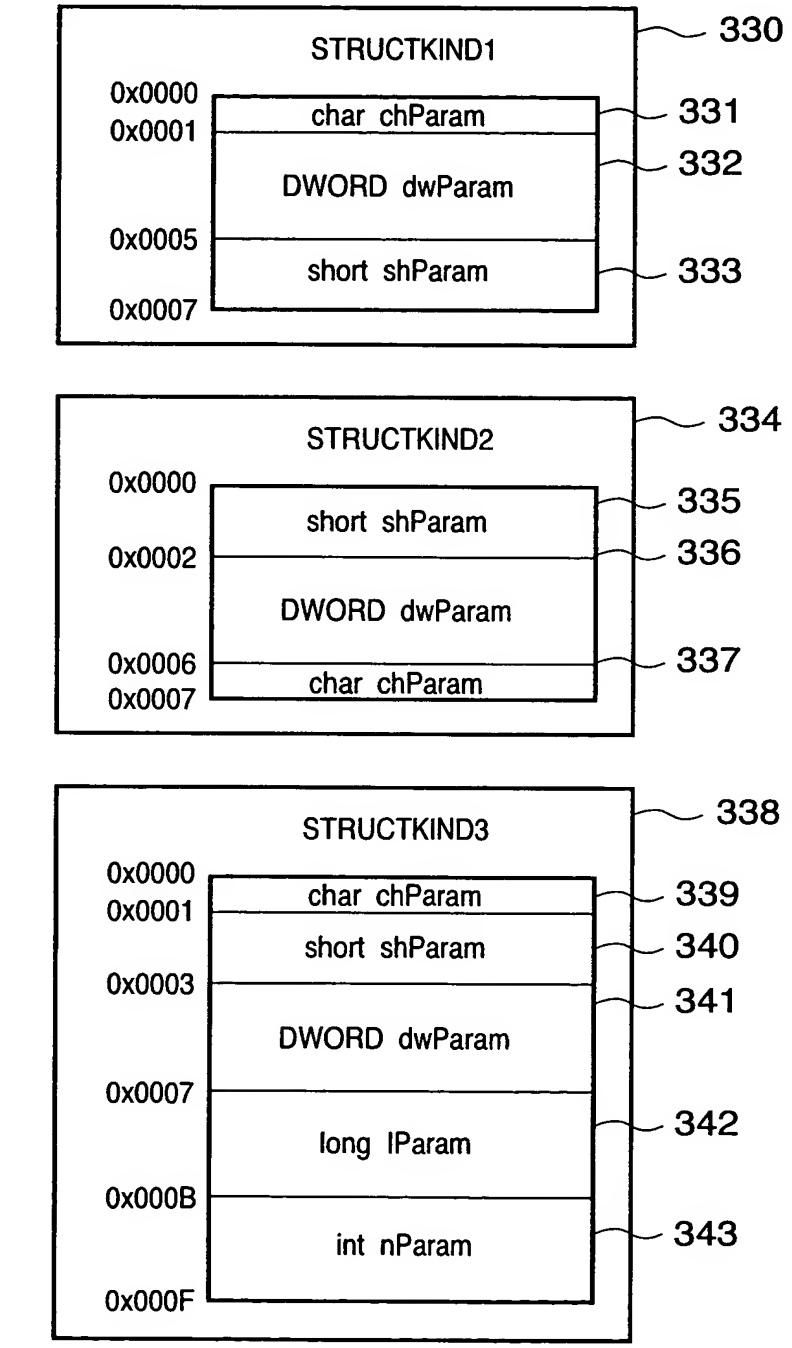
# FIG. 27

```
MODULE NAME :     TestDllStd. DLL
FUNCTION NAME :   FuncGetData
ARGUMENT(in) :    DWORD dwKind : 1
                  STRUCTSIZE* pBuf : 0x503860C
                  DWORD STRUCTSIZE. dwSize : 16
                  DWORD STRUCTSIZE. dwParam1 : 0
                  DWORD STRUCTSIZE. dwParam2 : 0
                  DWORD STRUCTSIZE. dwParam3 : 0
ARGUMENT(out) :   STRUCTSIZE* pBuf : 0x503860C
                  DWORD STRUCTSIZE. dwSize : 16
                  DWORD STRUCTSIZE. dwParam1 : 1
                  DWORD STRUCTSIZE. dwParam2 : 2
                  DWORD STRUCTSIZE. dwParam3 : 3
RETURN VALUE :    void :
IN TIME :         2002/03/25 22 : 24 : 12.025
OUT TIME :        2002/03/25 22 : 24 : 12.035

MODULE NAME :     TestDllStd. DLL
FUNCTION NAME :   FuncGetData
ARGUMENT(in) :    DWORD dwKind : 3
                  STRUCTSIZE* pBuf : 0x503990C
                  DWORD STRUCTSIZE. dwSize : 24
                  DWORD STRUCTSIZE. dwParam1 : 0
                  DWORD STRUCTSIZE. dwParam2 : 0
                  DWORD STRUCTSIZE. dwParam3 : 0
                  DWORD STRUCTSIZE. dwParam4 : 0
                  DWORD STRUCTSIZE. dwParam5 : 0
ARGUMENT(out) :   STRUCTSIZE* pBuf : 0x503990C
                  DWORD STRUCTSIZE. dwSize : 24
                  DWORD STRUCTSIZE. dwParam1 : 10
                  DWORD STRUCTSIZE. dwParam2 : 20
                  DWORD STRUCTSIZE. dwParam3 : 30
                  DWORD STRUCTSIZE. dwParam4 : 40
                  DWORD STRUCTSIZE. dwParam5 : 50
RETURN VALUE :    void :
IN TIME :         2002/03/25 22 : 24 : 12.046
OUT TIME :        2002/03/25 22 : 24 : 12.057

...
```

# FIG. 28

```
typedef struct
{
        char chParam;
        DWORD dwParam ;
        short shParam ;
}STRUCTKIND1 ;

typedef struct
{
        short shParam ;
        DWORD dwParam ;
        char chParam;
}STRUCTKIND2 ;

typedef struct
{
        char chParam;
        DWORD dwParam ;
        short shParam ;
        long lParam ;
        int nParam ;
}STRUCTKIND3 ;

void FuncGetData (DWORD dwKind, void* lpBuf)
{

        switch(dwKind)
        }
        case 1 :
                //lpBuf IS TREATED AS THE POINTER TO STRUCTKIND1
                break;
        case 2 :
                //lpBuf iS TREATED AS THE POINTER TO STRUCTKIND2
                break;
        case 3 :
                //lpBuf IS TREATED AS THE POINTER TO STRUCTKIND3
                break;
       }
}
```

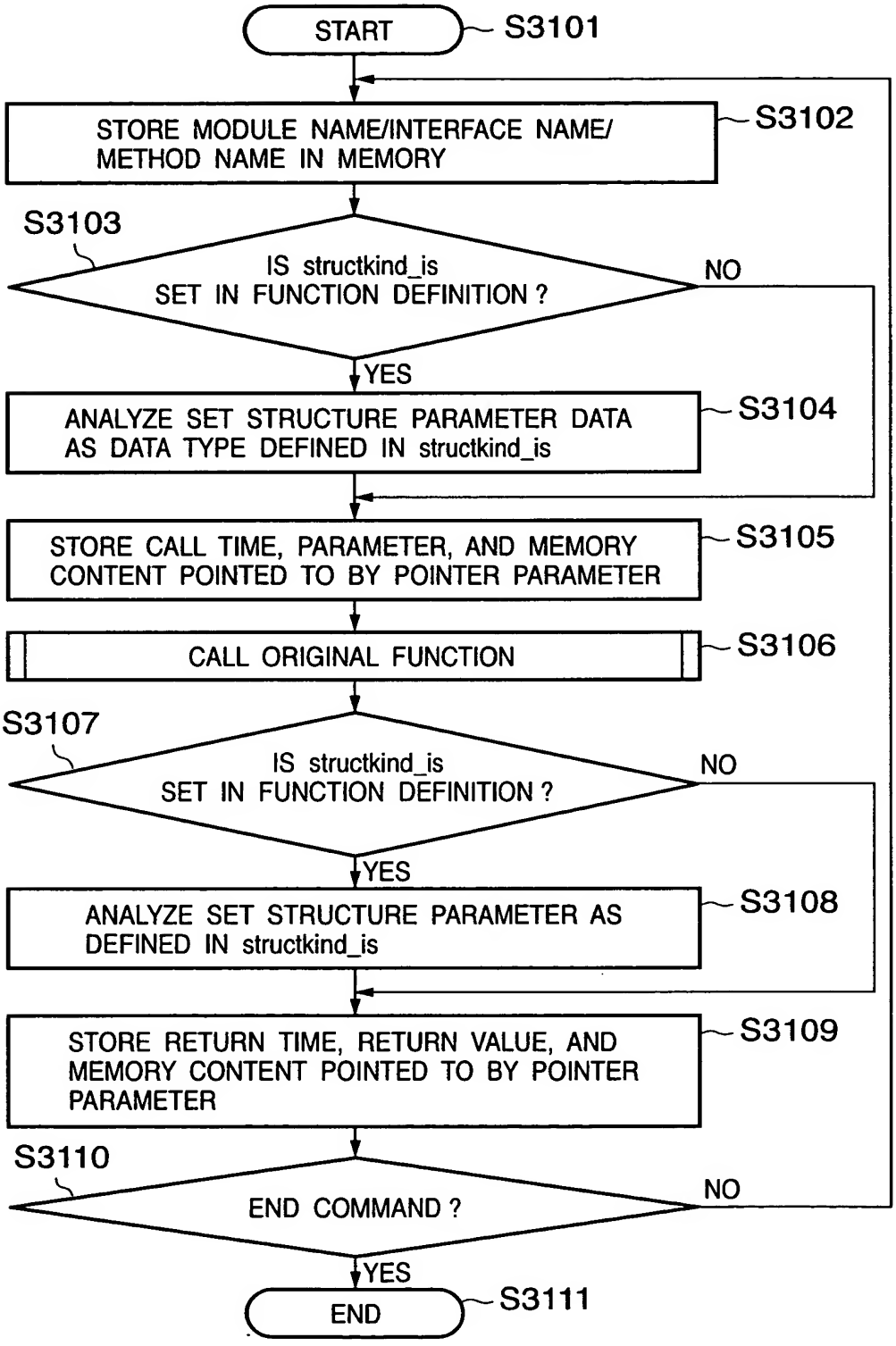
FIG. 29


STRUCTKIND1
330
0x0000
0x0001
char chParam
331
DWORD dwParam
332
0x0005
short shParam
333
0x0007
STRUCTKIND2
334
0x0000
short shParam
335
0x0002
DWORD dwParam
336
0x0006
0x0007
char chParam
337
STRUCTKIND3
338
0x0000
0x0001
char chParam
339
short shParam
340
0x0003
DWORD dwParam
341
0x0007
long lParam
342
0x000B
int nParam
343
0x000F

# FIG. 30

```
#define PAT_PARAM_ATTR_ID 00000000-0000-0000-000000000000

typedef [public] struct
{
        char chParam ;
        DWORD dwParam ;
        short shParam ;
}STRUCTKIND1 ;

typedef [public] struct
{
        short shParam ;
        DWORD dwParam ;
        char chParam ;
}STRUCTKIND2 ;

typedef [public] struct
{
        char chParam ;
        short shParam ;
        DWORD dwParam ;
        long lParam ;
        int nParam ;
}STRUCTKIND3 ;

interface
test
{
        void FuncGetData
        (
                [in] DWORD dwKind,
                [in, out, custom(PAT_PARAM_ATTR_ID,
                "structKind_is(dwKind : 1 : STRUCTKIND1*, 2 : STRUCTKIND2*, 3 : STRUCTKIND3* )")]
                void* lpBuf
        );

};
```

# FIG. 31


START
S3101
STORE MODULE NAME/INTERFACE NAME/ METHOD NAME IN MEMORY
S3102
S3103
IS structkind_is SET IN FUNCTION DEFINITION ?
NO
YES
ANALYZE SET STRUCTURE PARAMETER DATA AS DATA TYPE DEFINED IN structkind_is
S3104
STORE CALL TIME, PARAMETER, AND MEMORY CONTENT POINTED TO BY POINTER PARAMETER
S3105
CALL ORIGINAL FUNCTION
S3106
S3107
IS structkind_is SET IN FUNCTION DEFINITION ?
NO
YES
ANALYZE SET STRUCTURE PARAMETER AS DEFINED IN structkind_is
S3108
STORE RETURN TIME, RETURN VALUE, AND MEMORY CONTENT POINTED TO BY POINTER PARAMETER
S3109
S3110
END COMMAND ?
NO
YES
END
S3111

# FIG. 32

```
MODULE NAME :      TestDllStd. DLL
FUNCTION NAME :    FuncGetData
ARGUMENT(in) :     DWORD dwKind : 1
                   STRUCTKIND1* pBuf : 0x503860C
                   char STRUCTKIND1. chParam : 0
                   DWORD STRUCTKIND1. dwParam : 0
                   short STRUCTKIND1. shParam : 0
ARGUMENT(out) :    STRUCTKIND1* pBuf : 0x503860C
                   char STRUCTKIND1. chParam : 1
                   DWORD STRUCTKIND1. dwParam : 2
                   short STRUCTKIND1. shParam : 3

RETURN VALUE :     void :
IN TIME :          2002/03/25 22 : 24 : 12.025
OUT TIME :         2002/03/25 22 : 24 : 12.035

MODULE NAME :      TestDllStd. DLL
FUNCTION NAME :    FuncGetData
ARGUMENT(in) :     DWORD dwKind : 3
                   STRUCTKIND3* pBuf : 0x503990C
                   char STRUCTKIND3. chParam : 0
                   short STRUCTKIND3. shParam : 0
                   DWORD STRUCTKIND3. dwParam : 0
                   long STRUCTKIND3. lParam : 0
                   int STRUCTKIND3. nParam : 0
ARGUMENT(out) :    STRUCTKIND3* pBuf : 0x503990C
                   char STRUCTKIND3. chParam : 10
                   short STRUCTKIND3. shParam : 20
                   DWORD STRUCTKIND3. dwParam : 30
                   long STRUCTKIND3. lParam : 40
                   int STRUCTKIND3. nParam : 50
RETURN VALUE :     void :
IN TIME :          2002/03/25 22 : 24 : 12.046
OUT TIME :         2002/03/25 22 : 24 : 12.057

...
```

FIG. 33

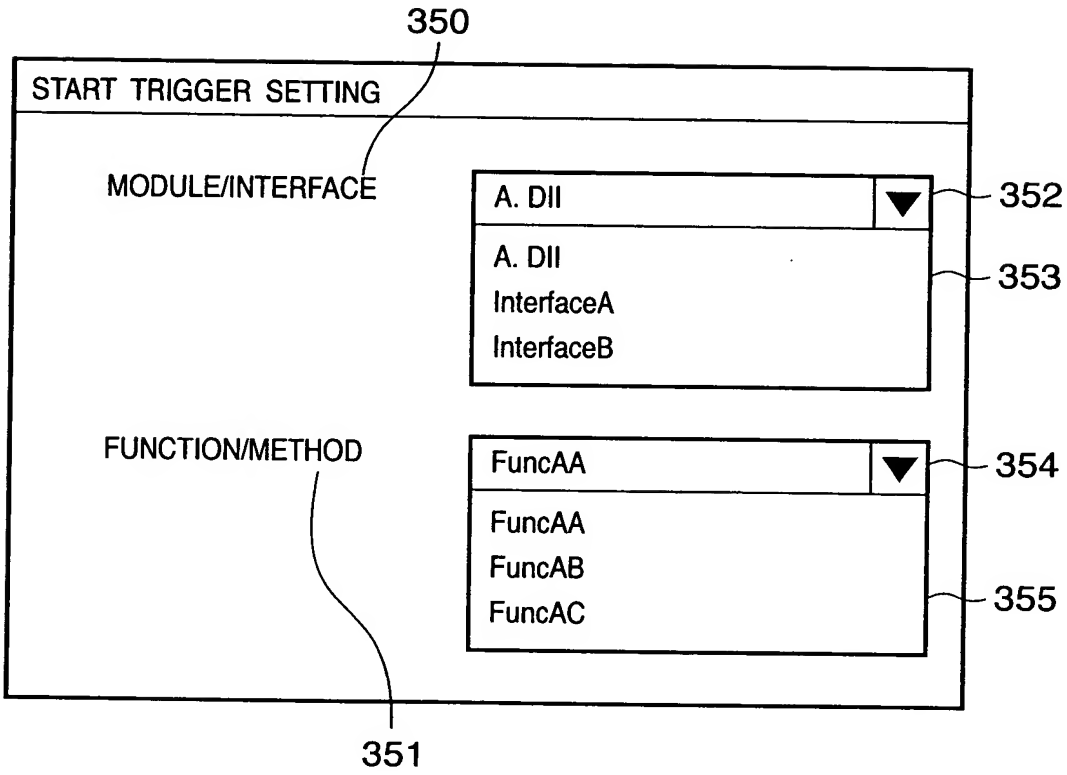

350
START TRIGGER SETTING
MODULE/INTERFACE
A. DII
352
A. DII
InterfaceA
InterfaceB
353
FUNCTION/METHOD
FuncAA
354
FuncAA
FuncAB
FuncAC
355
351

FIG. 34

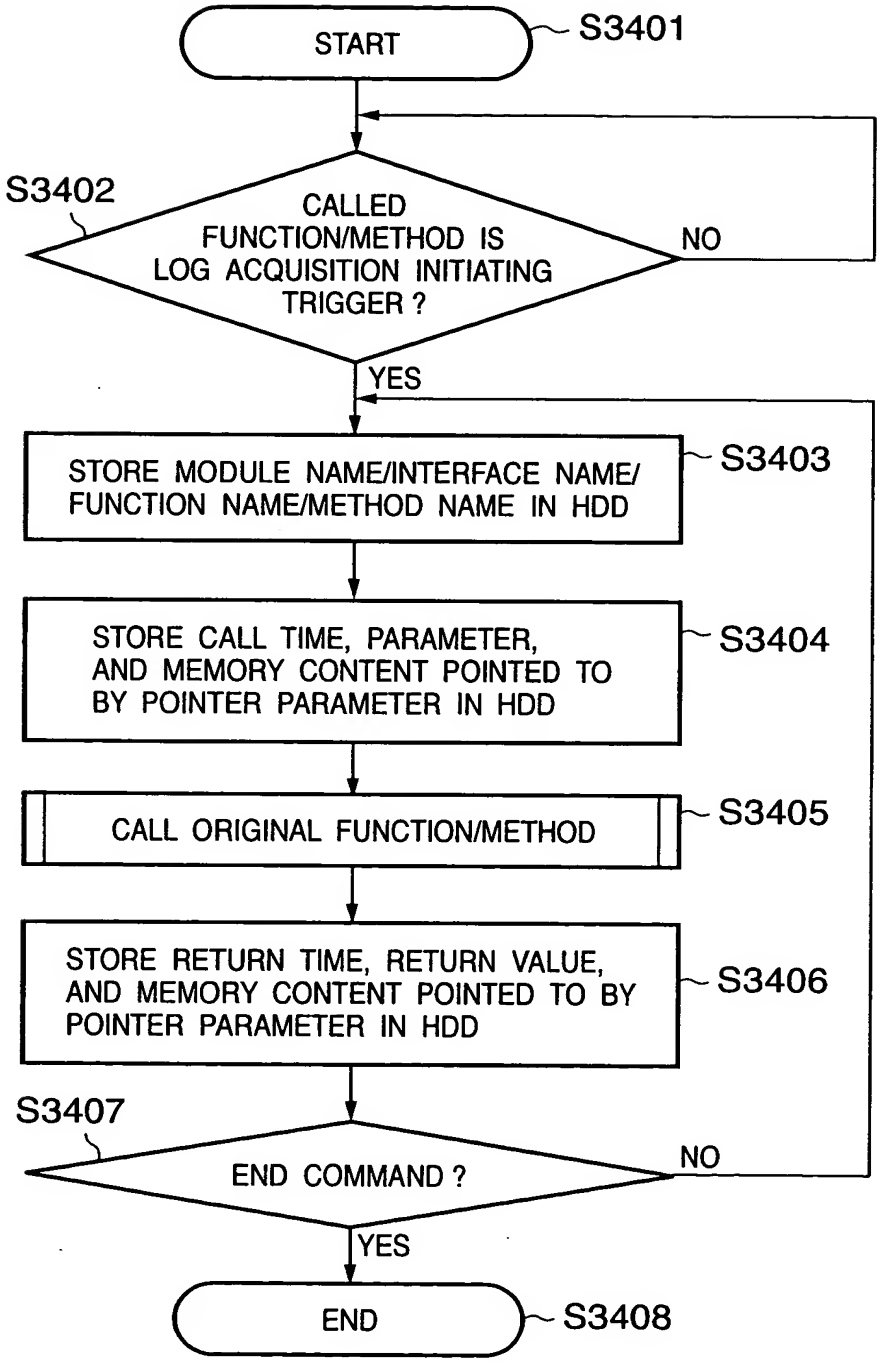

START
S3401
S3402
CALLED FUNCTION/METHOD IS LOG ACQUISITION INITIATING TRIGGER ?
NO
YES
STORE MODULE NAME/INTERFACE NAME/ FUNCTION NAME/METHOD NAME IN HDD
S3403
STORE CALL TIME, PARAMETER, AND MEMORY CONTENT POINTED TO BY POINTER PARAMETER IN HDD
S3404
CALL ORIGINAL FUNCTION/METHOD
S3405
STORE RETURN TIME, RETURN VALUE, AND MEMORY CONTENT POINTED TO BY POINTER PARAMETER IN HDD
S3406
S3407
END COMMAND ?
NO
YES
END
S3408

FIG. 35

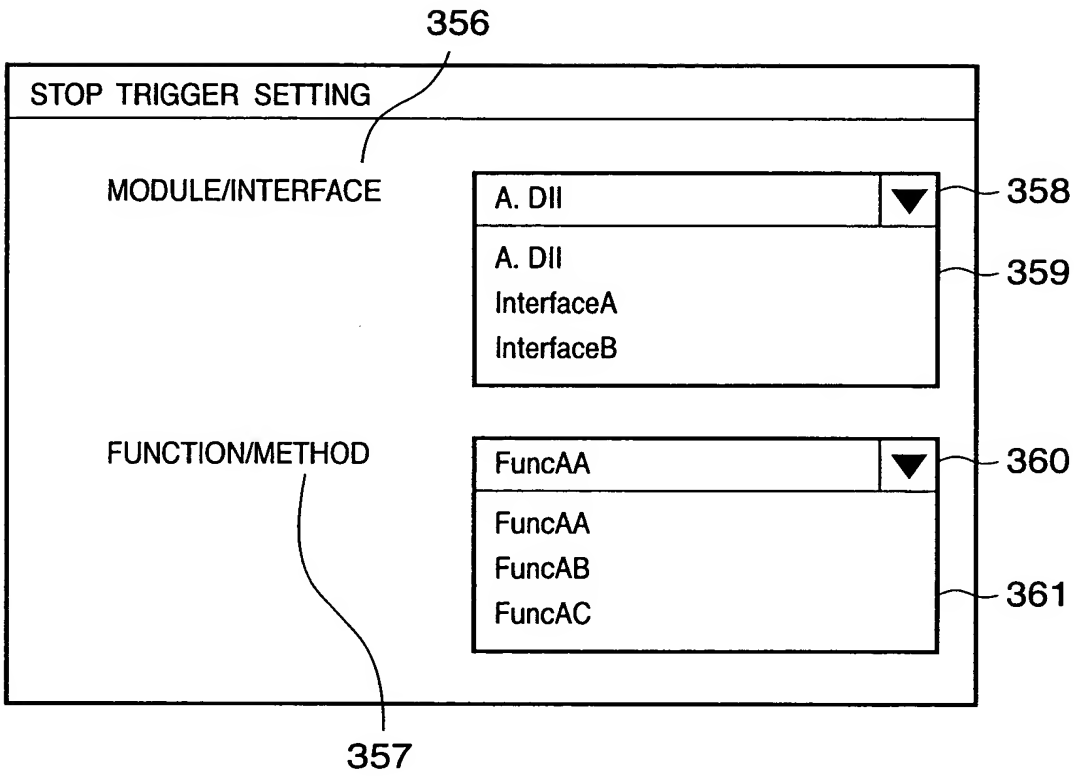

356
STOP TRIGGER SETTING
MODULE/INTERFACE
A. DII
A. DII
InterfaceA
InterfaceB
358
359
FUNCTION/METHOD
FuncAA
FuncAA
FuncAB
FuncAC
360
361
357

FIG. 36

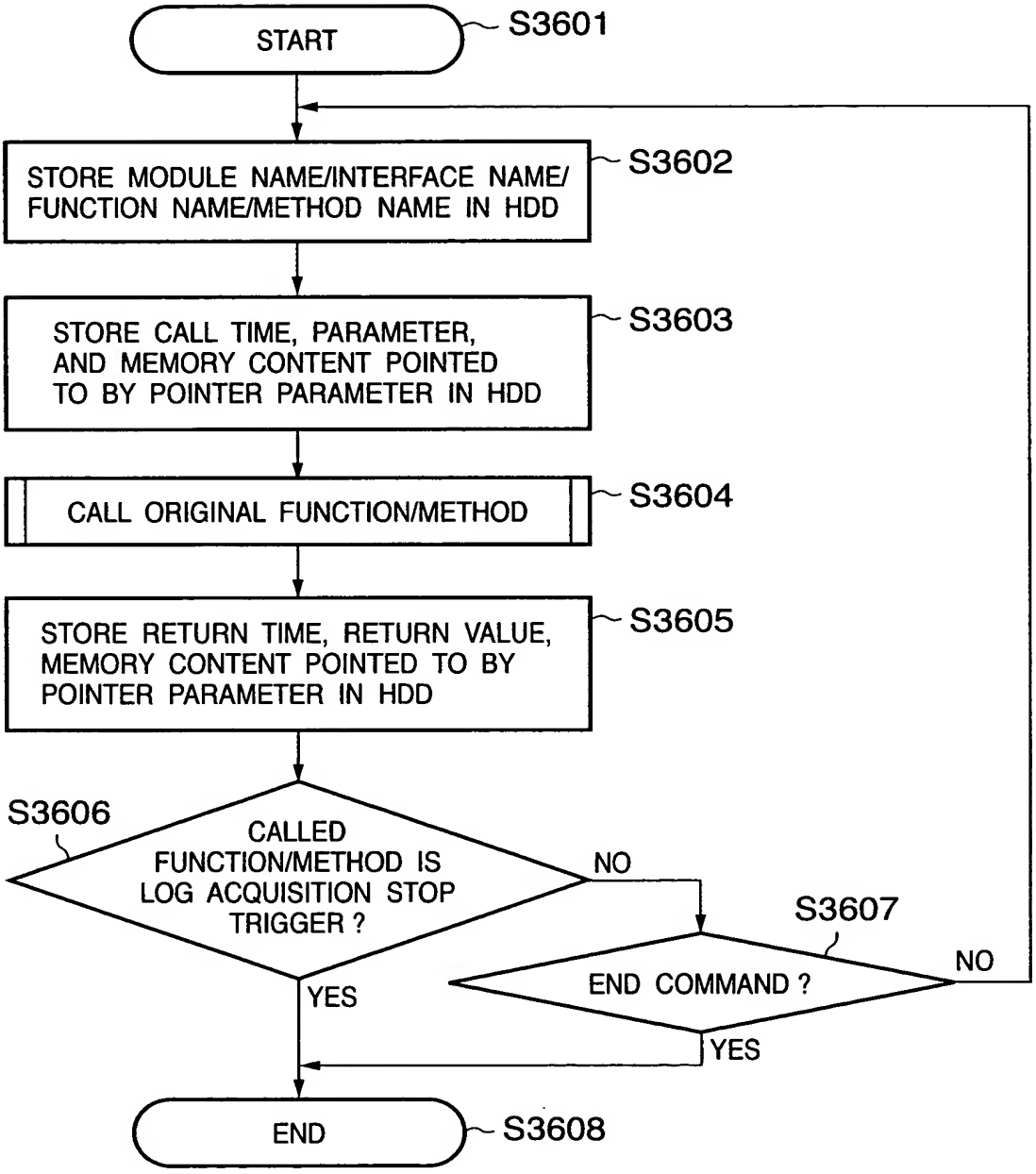
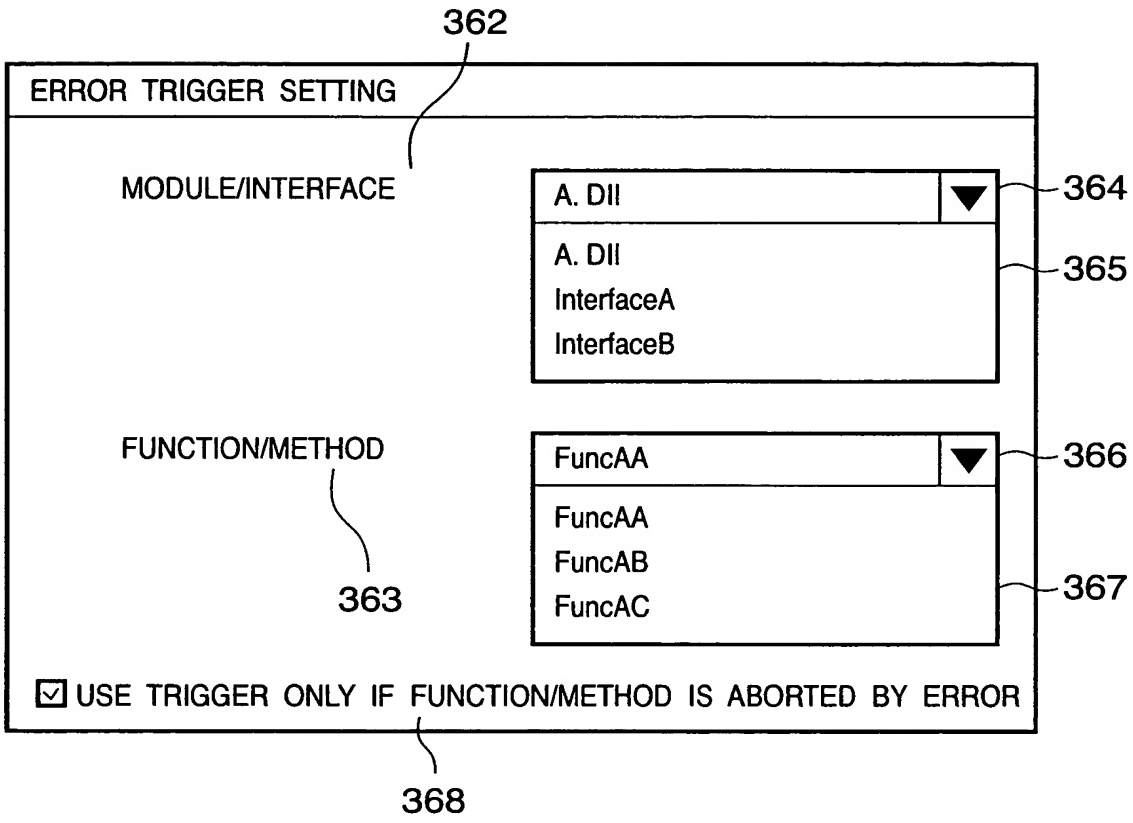

START
S3601
STORE MODULE NAME/INTERFACE NAME/ FUNCTION NAME/METHOD NAME IN HDD
S3602
STORE CALL TIME, PARAMETER, AND MEMORY CONTENT POINTED TO BY POINTER PARAMETER IN HDD
S3603
CALL ORIGINAL FUNCTION/METHOD
S3604
STORE RETURN TIME, RETURN VALUE, MEMORY CONTENT POINTED TO BY POINTER PARAMETER IN HDD
S3605
S3606
CALLED FUNCTION/METHOD IS LOG ACQUISITION STOP TRIGGER ?
NO
YES
S3607
END COMMAND ?
NO
YES
END
S3608

FIG. 37


362
ERROR TRIGGER SETTING
MODULE/INTERFACE
A. Dll
364
A. Dll
InterfaceA
InterfaceB
365
FUNCTION/METHOD
FuncAA
366
FuncAA
FuncAB
FuncAC
367
363
USE TRIGGER ONLY IF FUNCTION/METHOD IS ABORTED BY ERROR
368

# FIG. 38

| | |
|---|---|
| MODULE NAME : | A. DLL |
| FUNCTION NAME : | FuncAA |
| ARGUMENT : | DWORD dwID : Err>100 |
| RETURN VALUE : | DWORD dwRet : Err==0 |
| | |
| MODULE NAME : | A. DLL |
| FUNCTION NAME : | FuncAB |
| ARGUMENT : | DWORD dwHandle : Err==0 |
| RETURN VALUE : | int nRet : Err<=−1 |
| | |
| MODULE NAME : | B. DLL |
| INTERFACE NAME : | InterfaceA |
| METHOD NAME : | MethodAA |
| ARGUMENT : | DWORD dwID : Err>100 |
| RETURN VALUE : | DWORD dwHandle: Err==0 |
| | |
| MODULE NAME : | B. DLL |
| INTERFACE NAME : | InterfaceA |
| METHOD NAME : | MethodAB |
| ARGUMENT : | DWORD dwID : Err<=0 |
| RETURN VALUE : | DWORD dwRet: Err!=0 |
| | |
| MODULE NAME : | B. DLL |
| INTERFACE NAME : | InterfaceB |
| METHOD NAME : | MethodBA |
| ARGUMENT : | DWORD dwID : Err>=0 |
| RETURN VALUE : | DWORD dwRet: Err!=0 |

FIG. 39

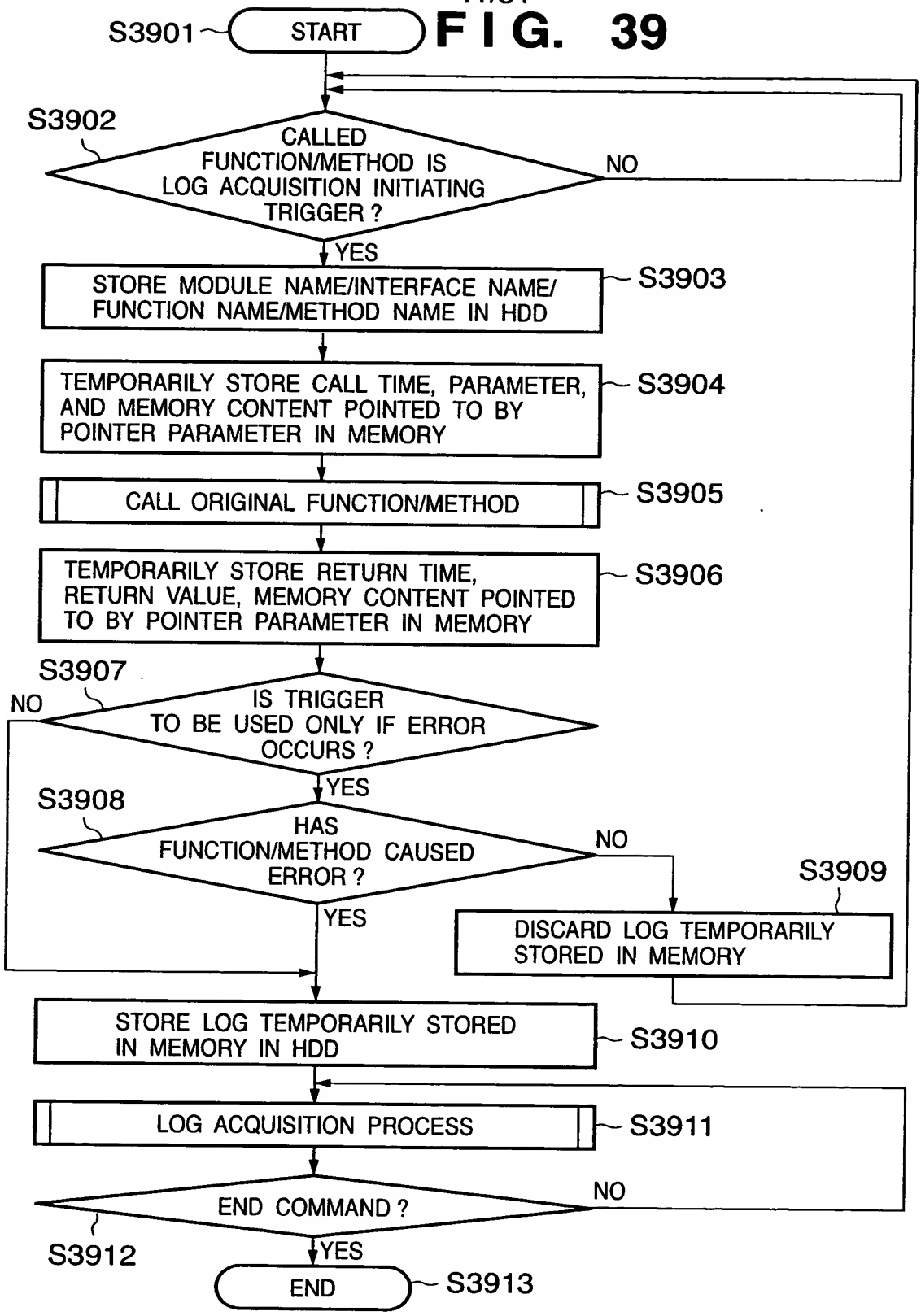

S3901
START
S3902
CALLED FUNCTION/METHOD IS LOG ACQUISITION INITIATING TRIGGER ?
NO
YES
STORE MODULE NAME/INTERFACE NAME/ FUNCTION NAME/METHOD NAME IN HDD
S3903
TEMPORARILY STORE CALL TIME, PARAMETER, AND MEMORY CONTENT POINTED TO BY POINTER PARAMETER IN MEMORY
S3904
CALL ORIGINAL FUNCTION/METHOD
S3905
TEMPORARILY STORE RETURN TIME, RETURN VALUE, MEMORY CONTENT POINTED TO BY POINTER PARAMETER IN MEMORY
S3906
S3907
NO
IS TRIGGER TO BE USED ONLY IF ERROR OCCURS ?
YES
S3908
HAS FUNCTION/METHOD CAUSED ERROR ?
NO
S3909
YES
DISCARD LOG TEMPORARILY STORED IN MEMORY
STORE LOG TEMPORARILY STORED IN MEMORY IN HDD
S3910
LOG ACQUISITION PROCESS
S3911
END COMMAND ?
NO
S3912
YES
END
S3913

FIG. 40

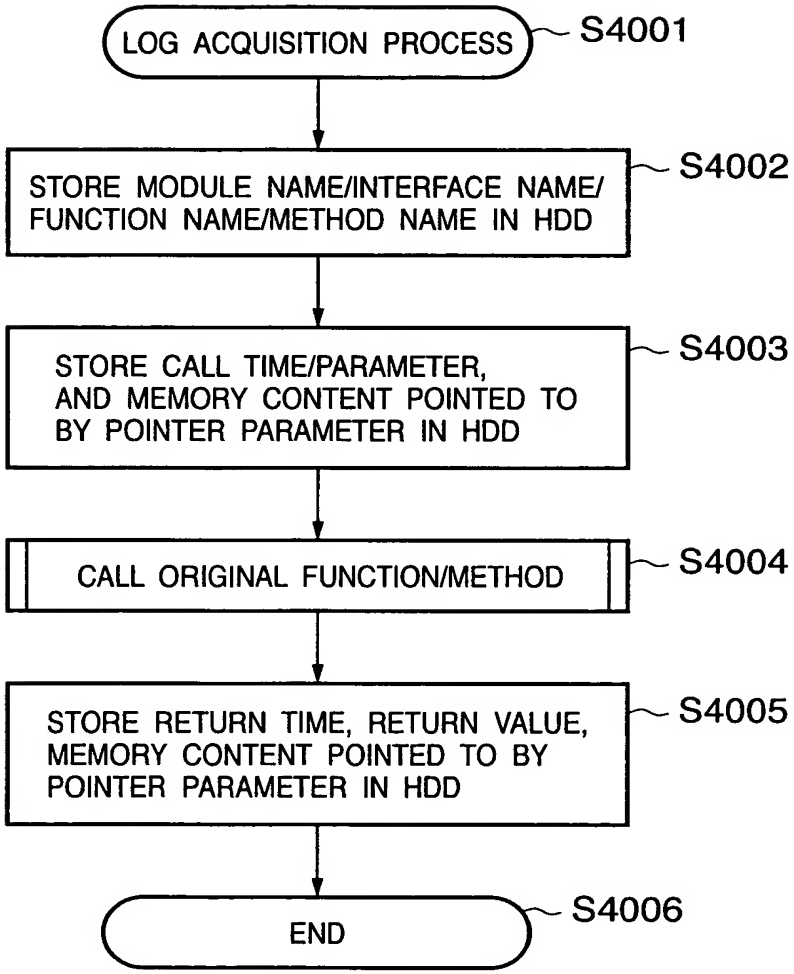

LOG ACQUISITION PROCESS
S4001
STORE MODULE NAME/INTERFACE NAME/ FUNCTION NAME/METHOD NAME IN HDD
S4002
STORE CALL TIME/PARAMETER, AND MEMORY CONTENT POINTED TO BY POINTER PARAMETER IN HDD
S4003
CALL ORIGINAL FUNCTION/METHOD
S4004
STORE RETURN TIME, RETURN VALUE, MEMORY CONTENT POINTED TO BY POINTER PARAMETER IN HDD
S4005
END
S4006

FIG. 41

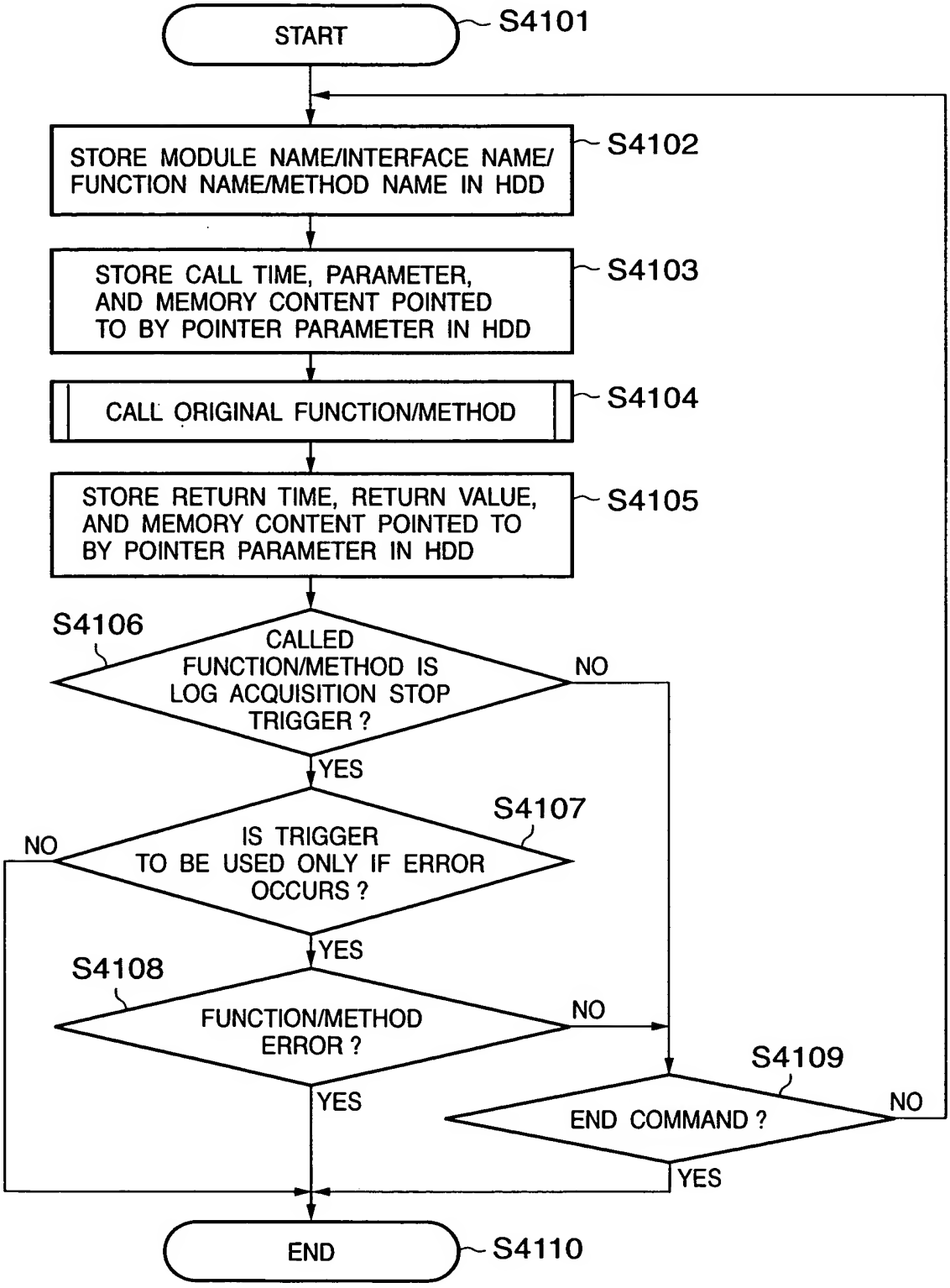

START
S4101
STORE MODULE NAME/INTERFACE NAME/ FUNCTION NAME/METHOD NAME IN HDD
S4102
STORE CALL TIME, PARAMETER, AND MEMORY CONTENT POINTED TO BY POINTER PARAMETER IN HDD
S4103
CALL ORIGINAL FUNCTION/METHOD
S4104
STORE RETURN TIME, RETURN VALUE, AND MEMORY CONTENT POINTED TO BY POINTER PARAMETER IN HDD
S4105
S4106
CALLED FUNCTION/METHOD IS LOG ACQUISITION STOP TRIGGER ?
NO
YES
S4107
IS TRIGGER TO BE USED ONLY IF ERROR OCCURS ?
NO
YES
S4108
FUNCTION/METHOD ERROR ?
NO
YES
S4109
END COMMAND ?
NO
YES
END
S4110

FIG. 42

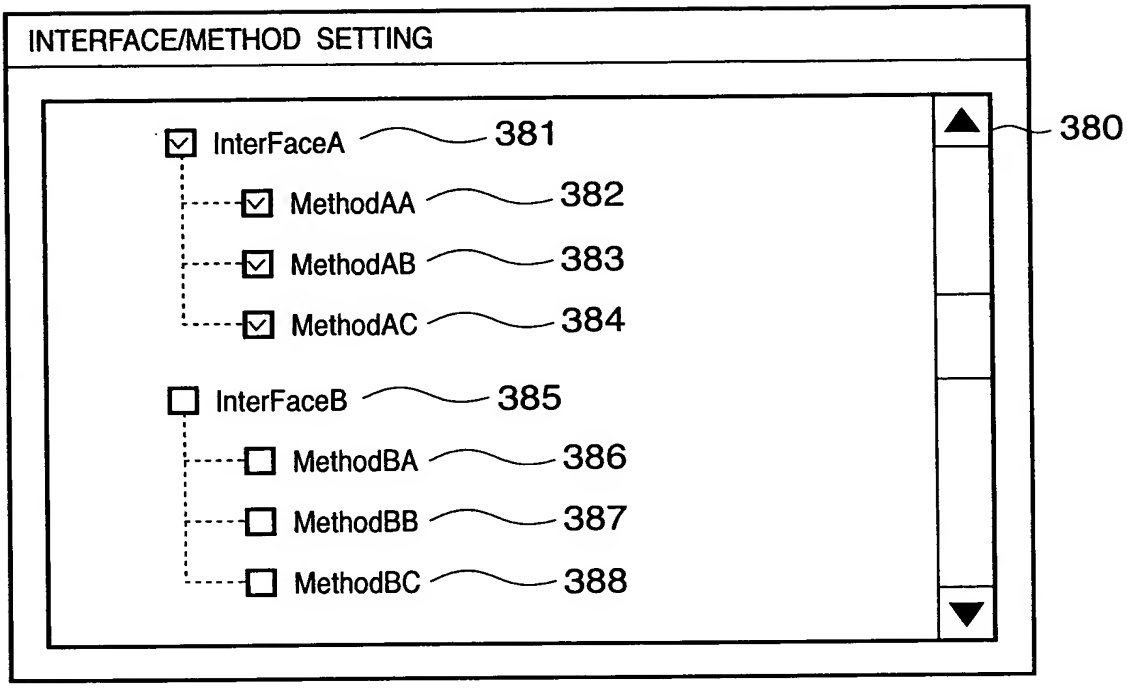

INTERFACE/METHOD SETTING
InterFaceA
381
MethodAA
382
MethodAB
383
MethodAC
384
InterFaceB
385
MethodBA
386
MethodBB
387
MethodBC
388
380

FIG. 43

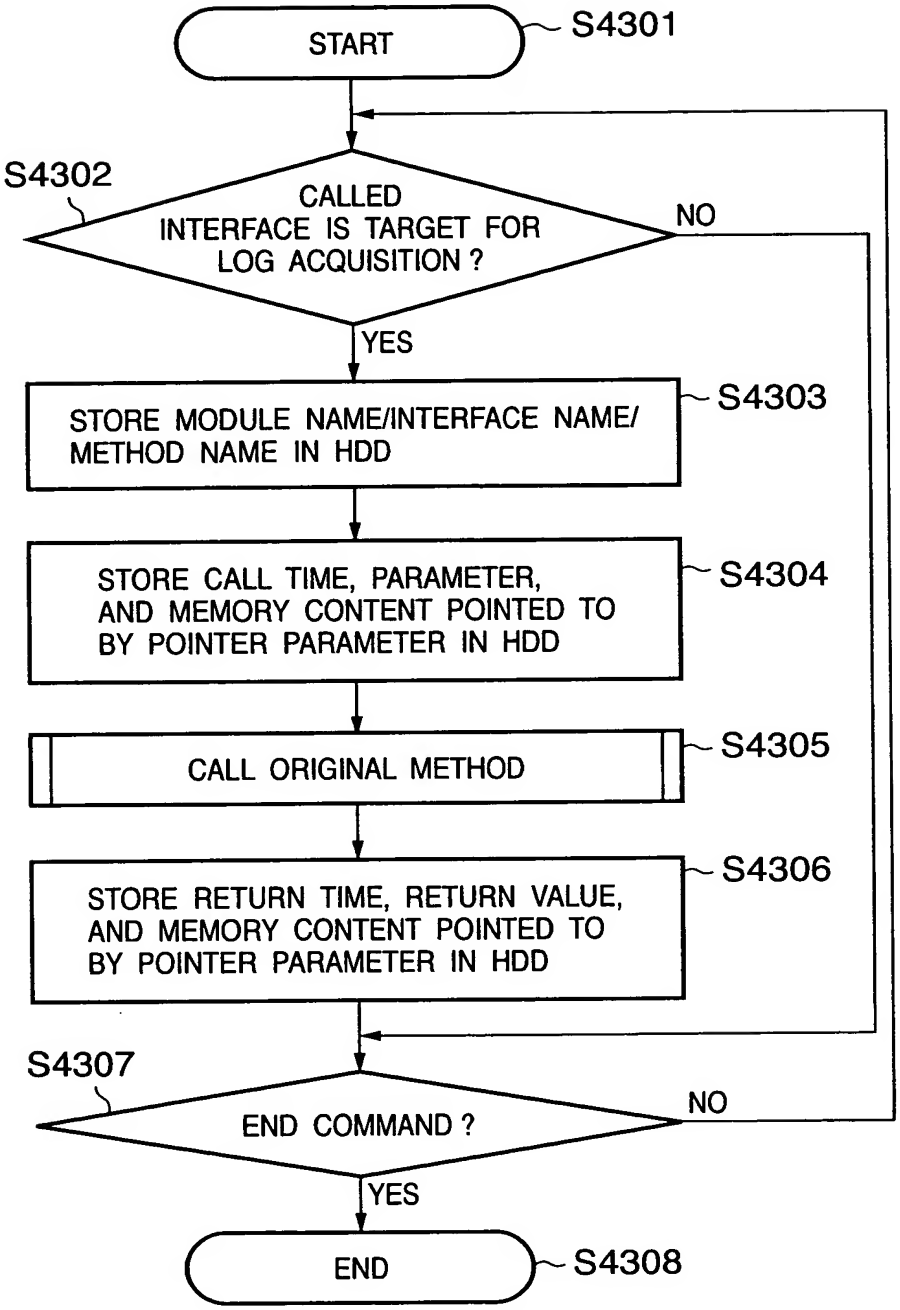

START
S4301
S4302
CALLED INTERFACE IS TARGET FOR LOG ACQUISITION ?
NO
YES
STORE MODULE NAME/INTERFACE NAME/ METHOD NAME IN HDD
S4303
STORE CALL TIME, PARAMETER, AND MEMORY CONTENT POINTED TO BY POINTER PARAMETER IN HDD
S4304
CALL ORIGINAL METHOD
S4305
STORE RETURN TIME, RETURN VALUE, AND MEMORY CONTENT POINTED TO BY POINTER PARAMETER IN HDD
S4306
S4307
END COMMAND ?
NO
YES
END
S4308

FIG. 44

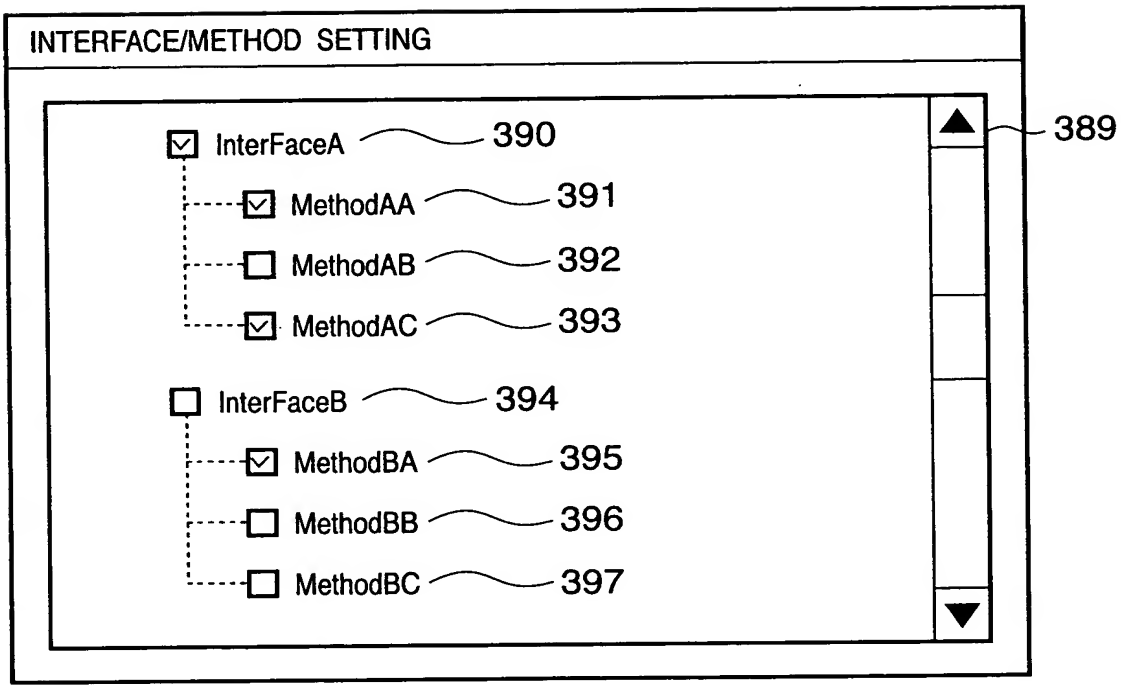

INTERFACE/METHOD SETTING
InterFaceA
390
MethodAA
391
MethodAB
392
MethodAC
393
InterFaceB
394
MethodBA
395
MethodBB
396
MethodBC
397
389

FIG. 45

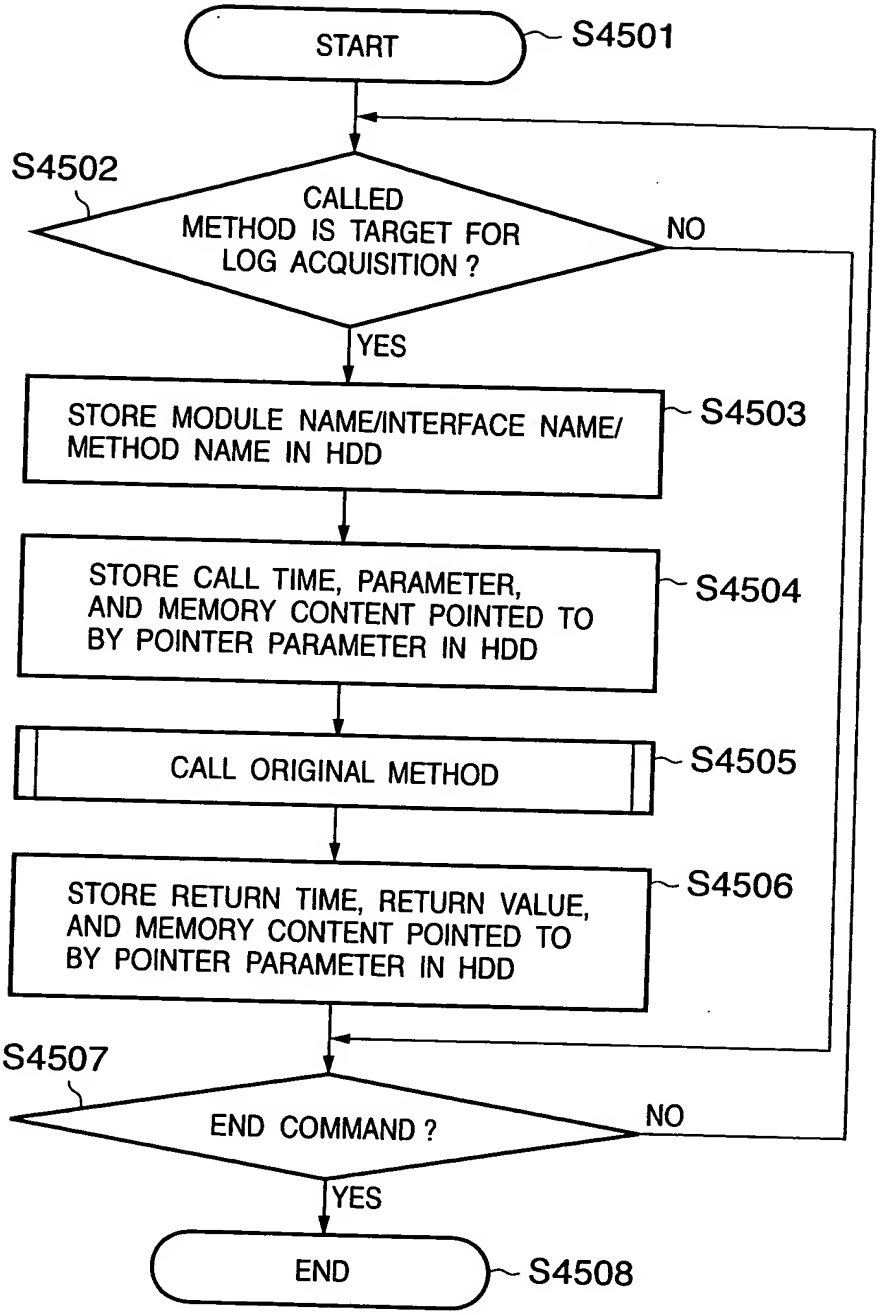

START
S4501
S4502
CALLED METHOD IS TARGET FOR LOG ACQUISITION ?
NO
YES
STORE MODULE NAME/INTERFACE NAME/ METHOD NAME IN HDD
S4503
STORE CALL TIME, PARAMETER, AND MEMORY CONTENT POINTED TO BY POINTER PARAMETER IN HDD
S4504
CALL ORIGINAL METHOD
S4505
STORE RETURN TIME, RETURN VALUE, AND MEMORY CONTENT POINTED TO BY POINTER PARAMETER IN HDD
S4506
S4507
END COMMAND ?
NO
YES
END
S4508

FIG. 46

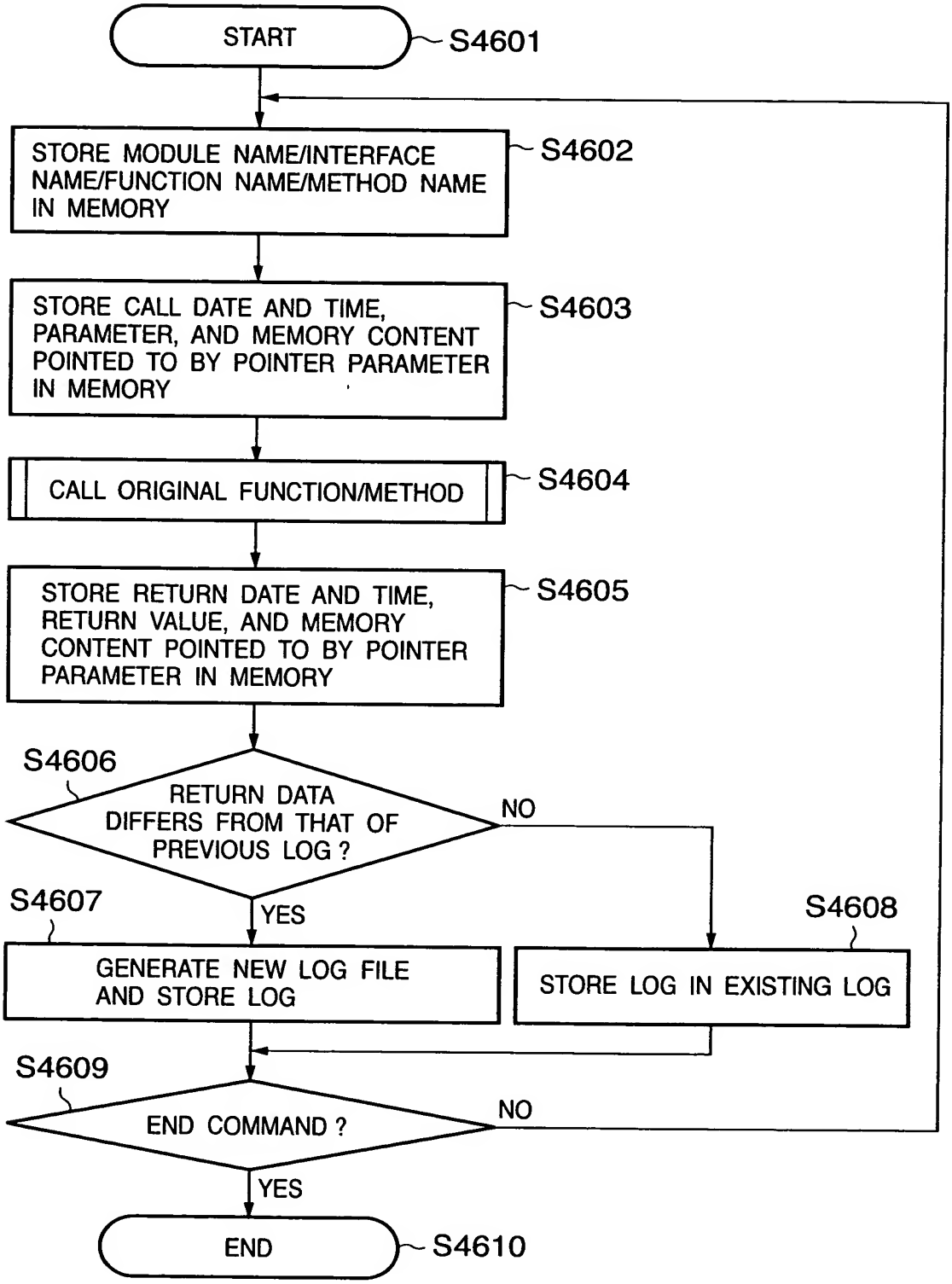

START
S4601
STORE MODULE NAME/INTERFACE NAME/FUNCTION NAME/METHOD NAME IN MEMORY
S4602
STORE CALL DATE AND TIME, PARAMETER, AND MEMORY CONTENT POINTED TO BY POINTER PARAMETER IN MEMORY
S4603
CALL ORIGINAL FUNCTION/METHOD
S4604
STORE RETURN DATE AND TIME, RETURN VALUE, AND MEMORY CONTENT POINTED TO BY POINTER PARAMETER IN MEMORY
S4605
S4606
RETURN DATA DIFFERS FROM THAT OF PREVIOUS LOG ?
NO
YES
S4607
GENERATE NEW LOG FILE AND STORE LOG
S4608
STORE LOG IN EXISTING LOG
S4609
END COMMAND ?
NO
YES
END
S4610

FIG. 47

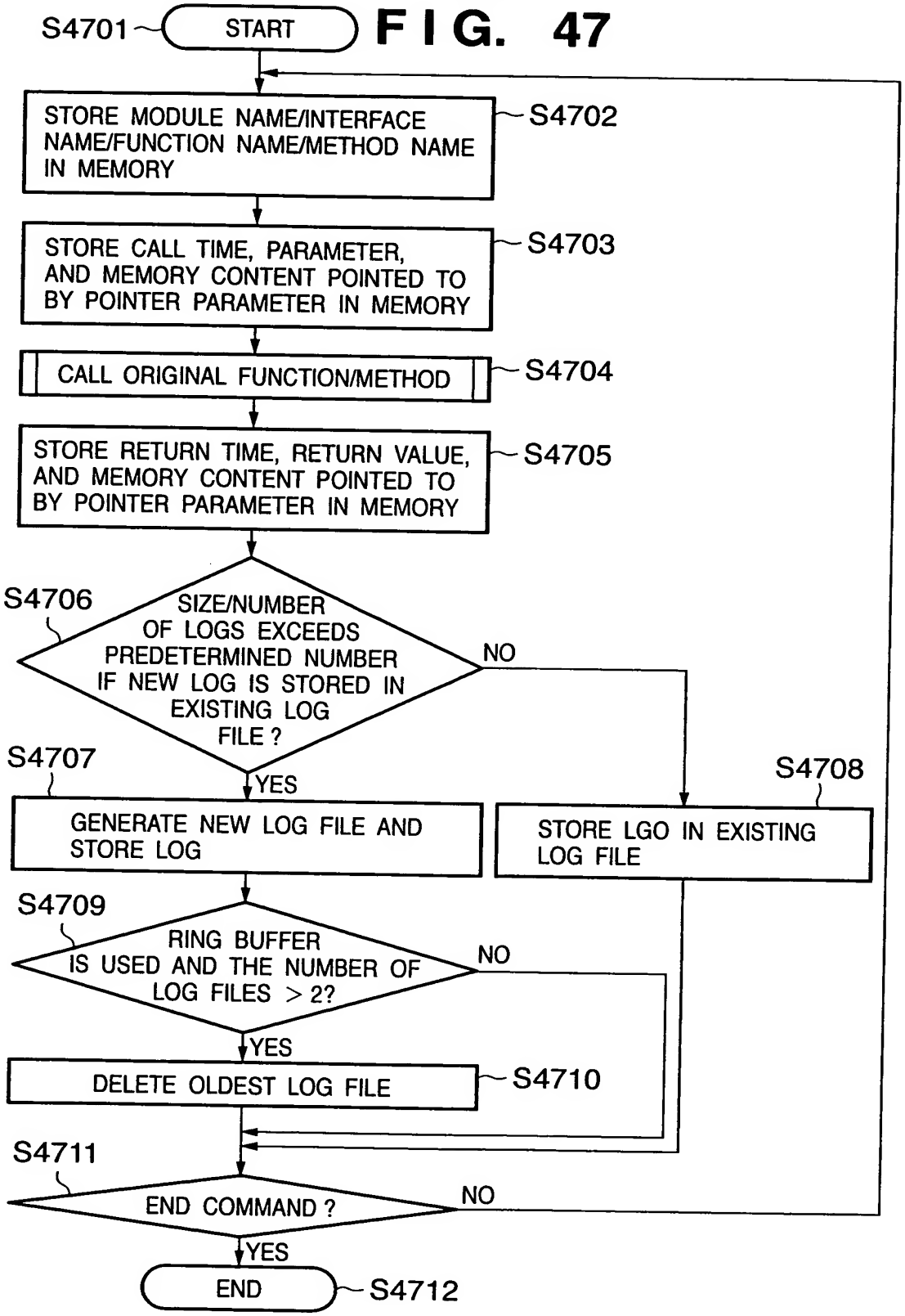

S4701
START
STORE MODULE NAME/INTERFACE NAME/FUNCTION NAME/METHOD NAME IN MEMORY
S4702
STORE CALL TIME, PARAMETER, AND MEMORY CONTENT POINTED TO BY POINTER PARAMETER IN MEMORY
S4703
CALL ORIGINAL FUNCTION/METHOD
S4704
STORE RETURN TIME, RETURN VALUE, AND MEMORY CONTENT POINTED TO BY POINTER PARAMETER IN MEMORY
S4705
S4706
SIZE/NUMBER OF LOGS EXCEEDS PREDETERMINED NUMBER IF NEW LOG IS STORED IN EXISTING LOG FILE ?
NO
YES
S4707
GENERATE NEW LOG FILE AND STORE LOG
S4708
STORE LGO IN EXISTING LOG FILE
S4709
RING BUFFER IS USED AND THE NUMBER OF LOG FILES > 2?
NO
YES
DELETE OLDEST LOG FILE
S4710
S4711
END COMMAND ?
NO
YES
END
S4712

FIG. 48

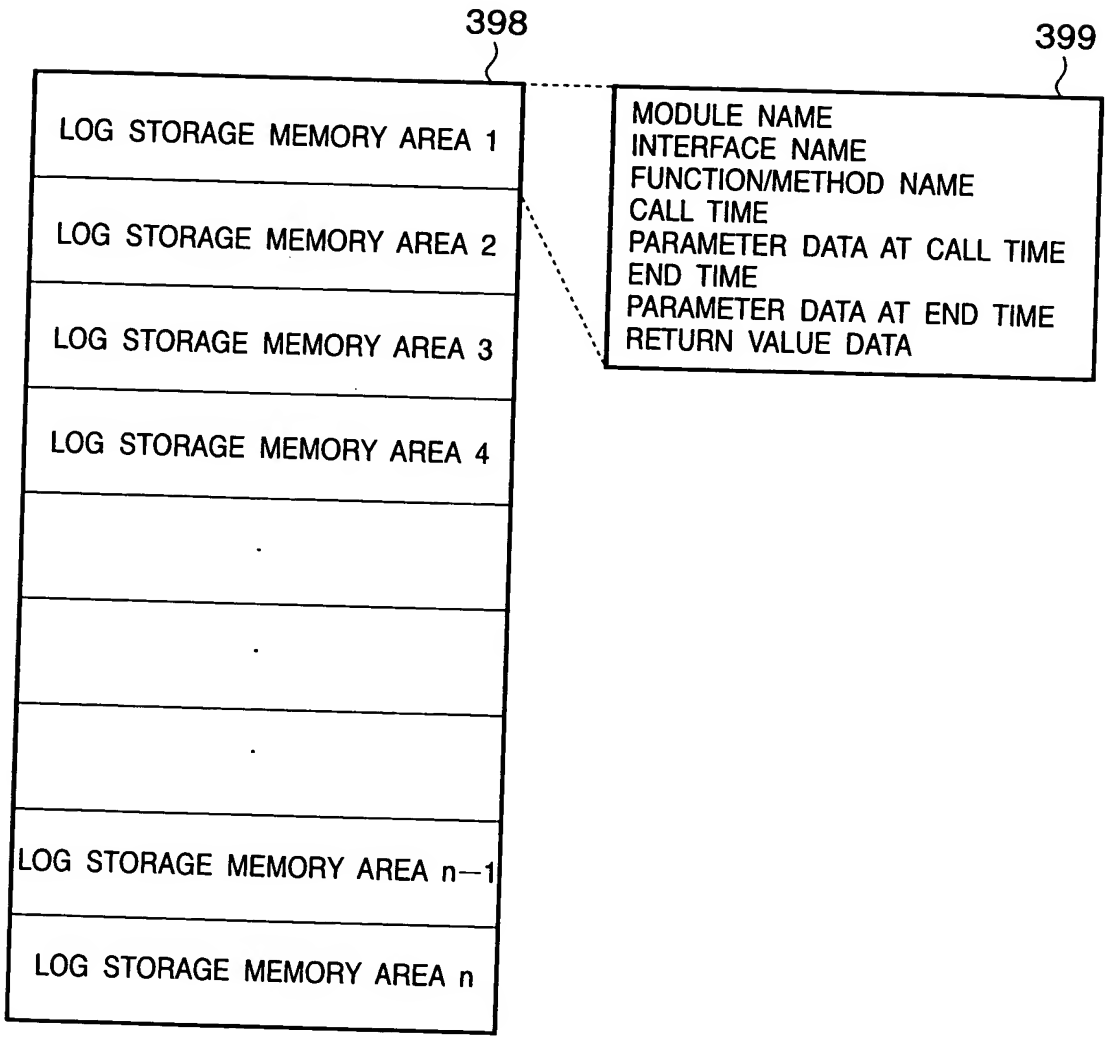
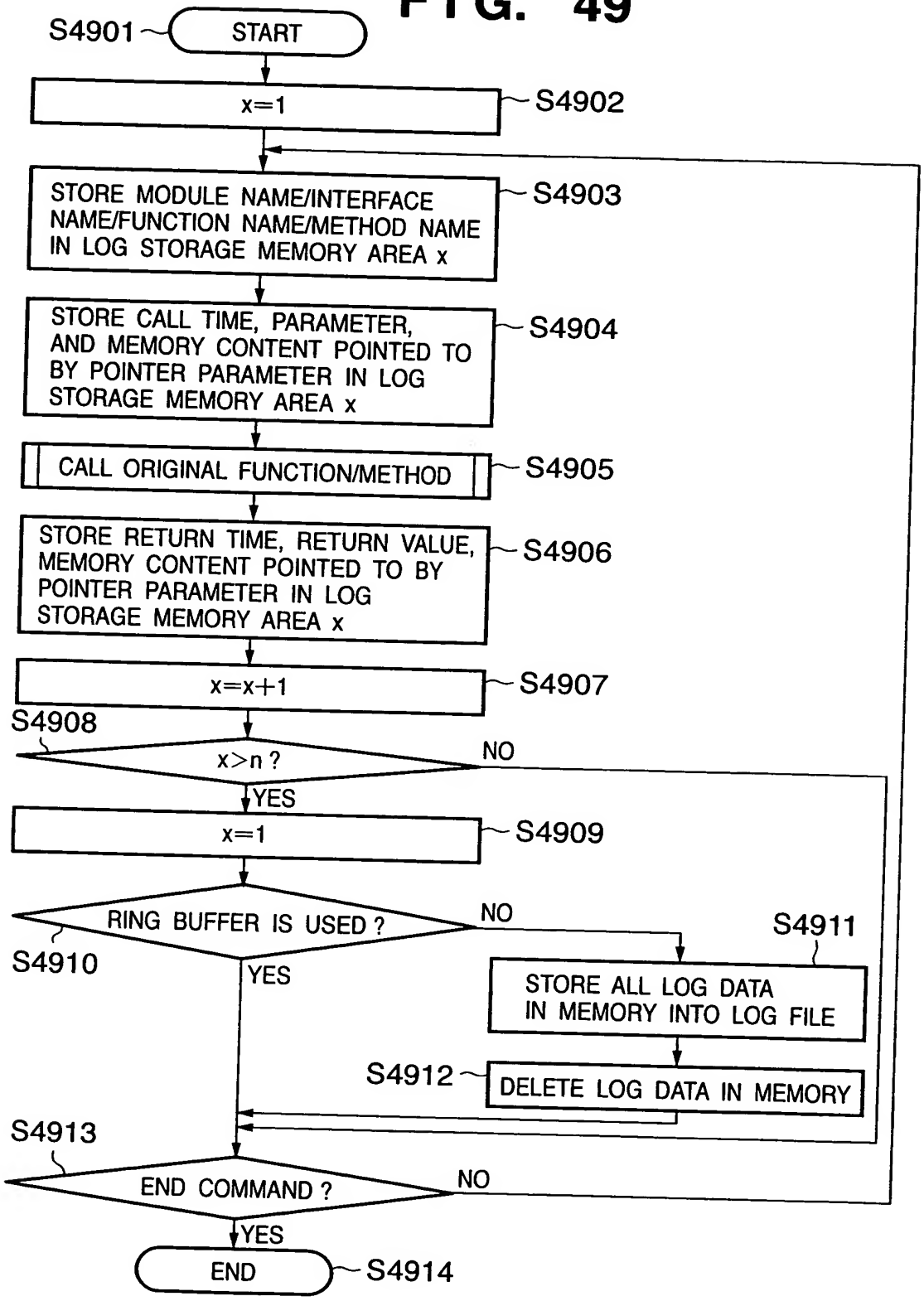

398
399
LOG STORAGE MEMORY AREA 1
LOG STORAGE MEMORY AREA 2
LOG STORAGE MEMORY AREA 3
LOG STORAGE MEMORY AREA 4
LOG STORAGE MEMORY AREA n−1
LOG STORAGE MEMORY AREA n
MODULE NAME
INTERFACE NAME
FUNCTION/METHOD NAME
CALL TIME
PARAMETER DATA AT CALL TIME
END TIME
PARAMETER DATA AT END TIME
RETURN VALUE DATA

FIG. 49


S4901
START
x=1
S4902
STORE MODULE NAME/INTERFACE NAME/FUNCTION NAME/METHOD NAME IN LOG STORAGE MEMORY AREA x
S4903
STORE CALL TIME, PARAMETER, AND MEMORY CONTENT POINTED TO BY POINTER PARAMETER IN LOG STORAGE MEMORY AREA x
S4904
CALL ORIGINAL FUNCTION/METHOD
S4905
STORE RETURN TIME, RETURN VALUE, MEMORY CONTENT POINTED TO BY POINTER PARAMETER IN LOG STORAGE MEMORY AREA x
S4906
x=x+1
S4907
S4908
x>n ?
NO
YES
x=1
S4909
RING BUFFER IS USED ?
NO
S4911
S4910
YES
STORE ALL LOG DATA IN MEMORY INTO LOG FILE
S4912
DELETE LOG DATA IN MEMORY
S4913
END COMMAND ?
NO
YES
END
S4914